Introduction

2024.03.07

SWPP Practice Session

Seunghyeon Nam (with lots of derived works)

About practice session

- Software Development Principles & Practices
- Covers more practical issues related to actual development
- Thursday 19:00 ~ 21:00 (KST), face-to-face session
- No attendance check, but come for your own benefit :)

Schedules (tentative)

- Week 1: Practice Session Intro & Setup & Git Tutorial
- Week 2: C++ Primer
- Week 3~?: LLVM and IR
- Early April?~: Project introduction, collaboration, and tips

Sign Up for GitHub

- A web service for collaborative development

  https://github.com

- Create GitHub account and submit your username by 3/11!

  More details in this GitHub issue

- Announcements and updates will be posted on GitHub Issues
  - They will NOT be posted on eTL!

Development Environment

- Use Linux or macOS
- If you're new to Linux, try Ubuntu Desktop.

  Download Ubuntu Desktop
- Or, use WSL*Windows Subsystem for Linux* if you use Windows 10/11.

  Official WSL installation guide
- macOS users: Disable iCloud sync for your project directories!

Development Environment

- We recommend using Visual Studio Code

  Download Visual Studio Code

  - Lightweight and portable (Windows, macOS, Linux, x86, ARM, …)
  - Integrated git and GitHub functionalities
  - Vast amount of extensions
  - ~~Quicker response from TA~~

Development Environment

- Useful extensions for Visual Studio Code
  - clangd: Syntax highlighting, type hint, error squiggle, autocomplete, formatting, jump to file, and many more!
  - Cmake & CMake Tools: Quick configuration, build shortcuts
  - LLVM: LLVM IR syntax highlighting

Development Environment

- Use Remote extensions for remote server or WSL
  - Remote – SSH for servers connected through SSH
  - Remote – WSL ‘connects’ to Linux subsystem from Windows
  - Most extensions can be installed on remote side as well

Development Environment

- We will use clang++ 18.1.0 throughout the semester
- LLVM and project skeletons use CMake

  [Download CMake]

- Using Ninja is recommended for faster build

  [Download Ninja]

- You can also get CMake and Ninja via package managers

Development Environment

- We’ll use LLVM throughout this semester
  - Most assignments are about LLVM
  - Term project is based on LLVM
- Try building LLVM from source on your own!
  - First try getting used to CLI$_{\textit{command–line interface}}$ if you’re not familiar with it
  - Also, check if your development environment is well–configured

Helper Scripts

- install-llvm.sh
  - Start from this script if you’re not familiar with build systems
  - Downloads and installs LLVM along with its dependencies
  - swpp202401/practice-materials/scripts-〈OS〉/install-llvm.sh

Helper Scripts

- install-cmake.sh, install-ninja.sh
  - Older versions of build tools do not support new language features
  - Scripts will download and install latest tools and its dependencies
  - swpp202401/practice-materials/scripts-<OS>/install-cmake.sh
  - swpp202401/practice-materials/scripts-linux/install-ninja.sh
    - For MacOS, Brew already provides the latest Ninja

Testing your Development Environment

- dev-env-check
  - Checks if you have properly set up development environment
    - The latest LLVM, CMake, and Ninja
  - Written in & uses latest features in C++23
  - Please read README.md for more details

How to Ask Google

- Google always have answers
  - Well… almost always
  - But you have to 'properly' ask them
  - Coming up with good questions actually saves your time and energy!

This is such a sad story
O panzar of the lake, what is your wisdon
They always ask what my wisdom is, never how my wisdom is

How to Ask Google

- DO: ask in short noun form
  - linux download file from url
  - adding object to c++ vector
- DON'T: come up with full sentence
  - How can I download files from url in linux terminal?
  - I want to add an object to a c++ vector

How to Ask Google

- DO: ask about error message ‘templates’
  - error: invalid use of ‘void’
  - error: binding reference of type [omit!] discards qualifiers
- DO: ask about library objects, functions, etc
  - llvm::PassManager
  - std::accumulate

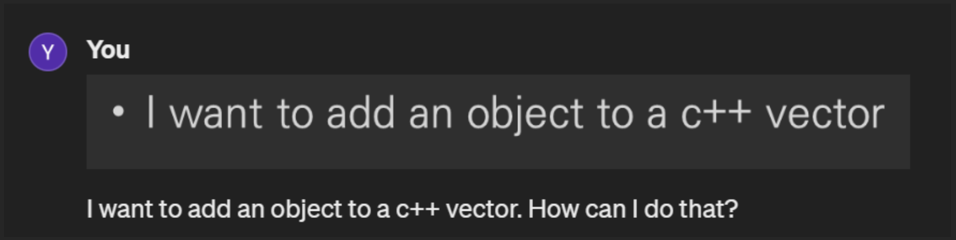
How to Ask Google

- DON’T: include your object/function name
  - error: binding reference of type ‘result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >&&’ to ‘std::remove_reference<const result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >&>::type’ {aka ‘const result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >’} discards qualifiers
  - Unfortunately, this isn’t trivial in C++ due to complex template substitution rules

How to Ask Google

- DO: put the programming language name at the front
  - c++ int to float
  - python int to float
- DON’T: omit the language name
  - int to float → c++? JavaScript? LLVM IR?

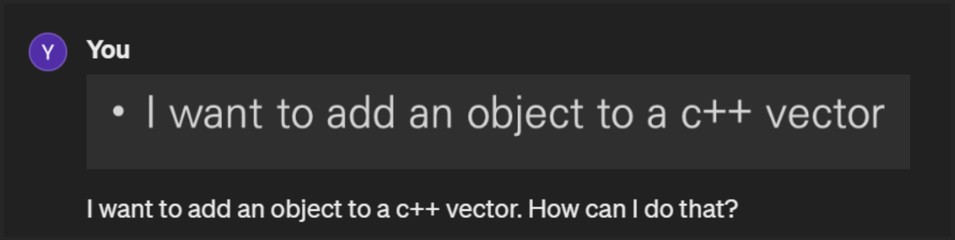
How to Ask ChatGPT



- I want to add an object to a c++ vector

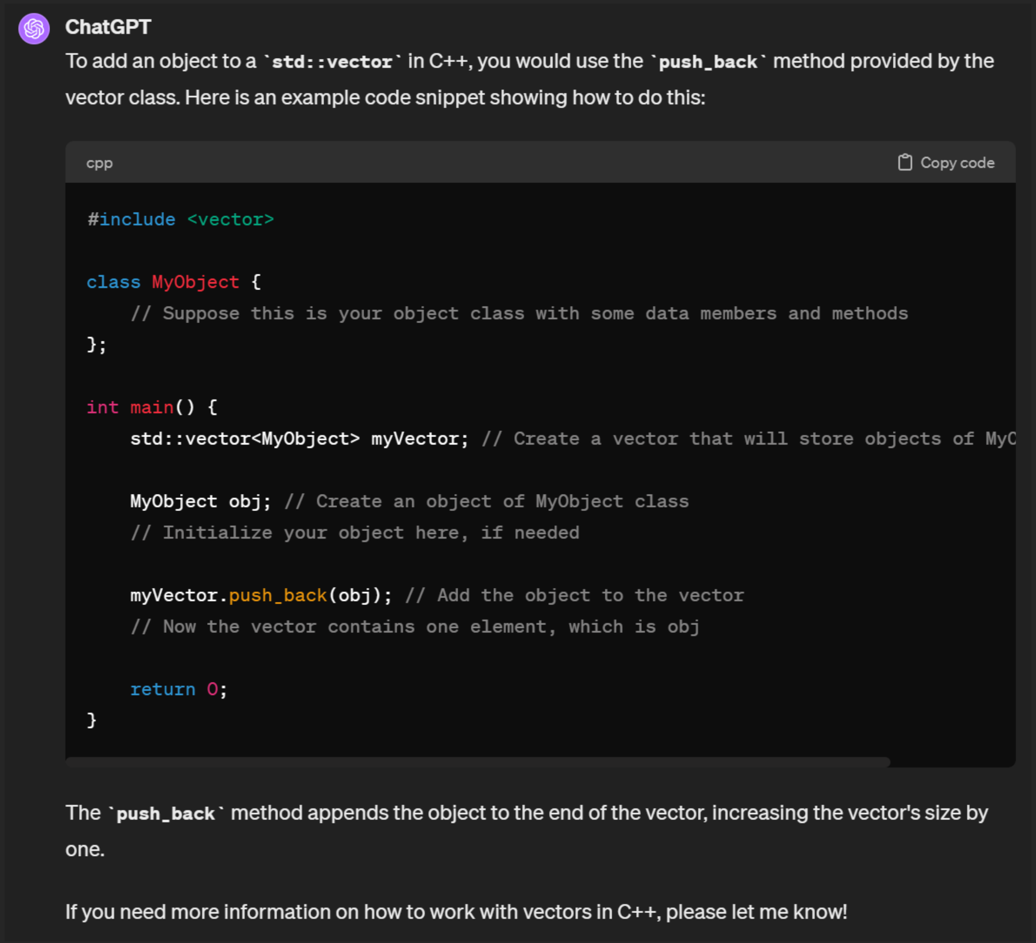
I want to add an object to a c++ vector. How can I do that?

How to Ask ChatGPT



- I want to add an object to a c++ vector

I want to add an object to a c++ vector. How can I do that?



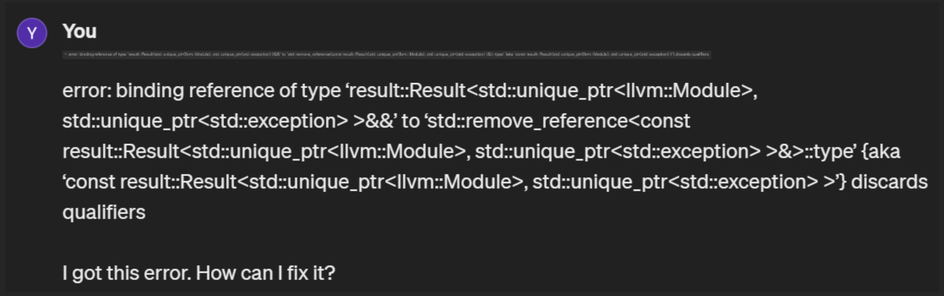
To add an object to a `std::vector` in C++, you would use the `push_back` method provided by the vector class. Here is an example code snippet showing how to do this:

```cpp
#include <vector>

class MyObject {
    // Suppose this is your object class with some data members and methods
};

int main() {
    std::vector<MyObject> myVector; // Create a vector that will store objects of MyC

    MyObject obj; // Create an object of MyObject class
    // Initialize your object here, if needed

    myVector.push_back(obj); // Add the object to the vector
    // Now the vector contains one element, which is obj

    return 0;
}
```

The `push_back` method appends the object to the end of the vector, increasing the vector's size by one.

If you need more information on how to work with vectors in C++, please let me know!

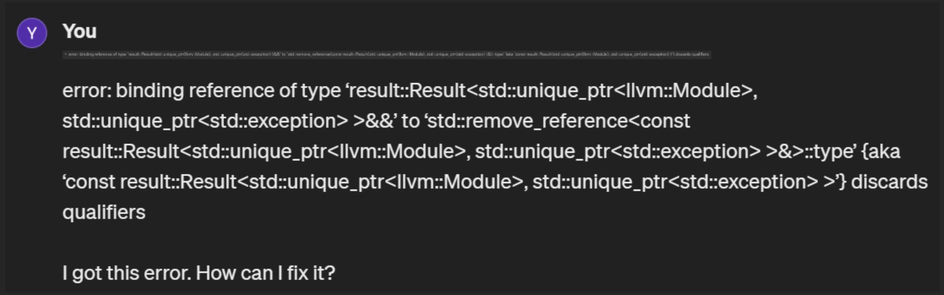
How to Ask ChatGPT



error: binding reference of type ‘result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >&&’ to ‘std::remove_reference<const result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >&>::type’ {aka ‘const result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >’} discards qualifiers

I got this error. How can I fix it?

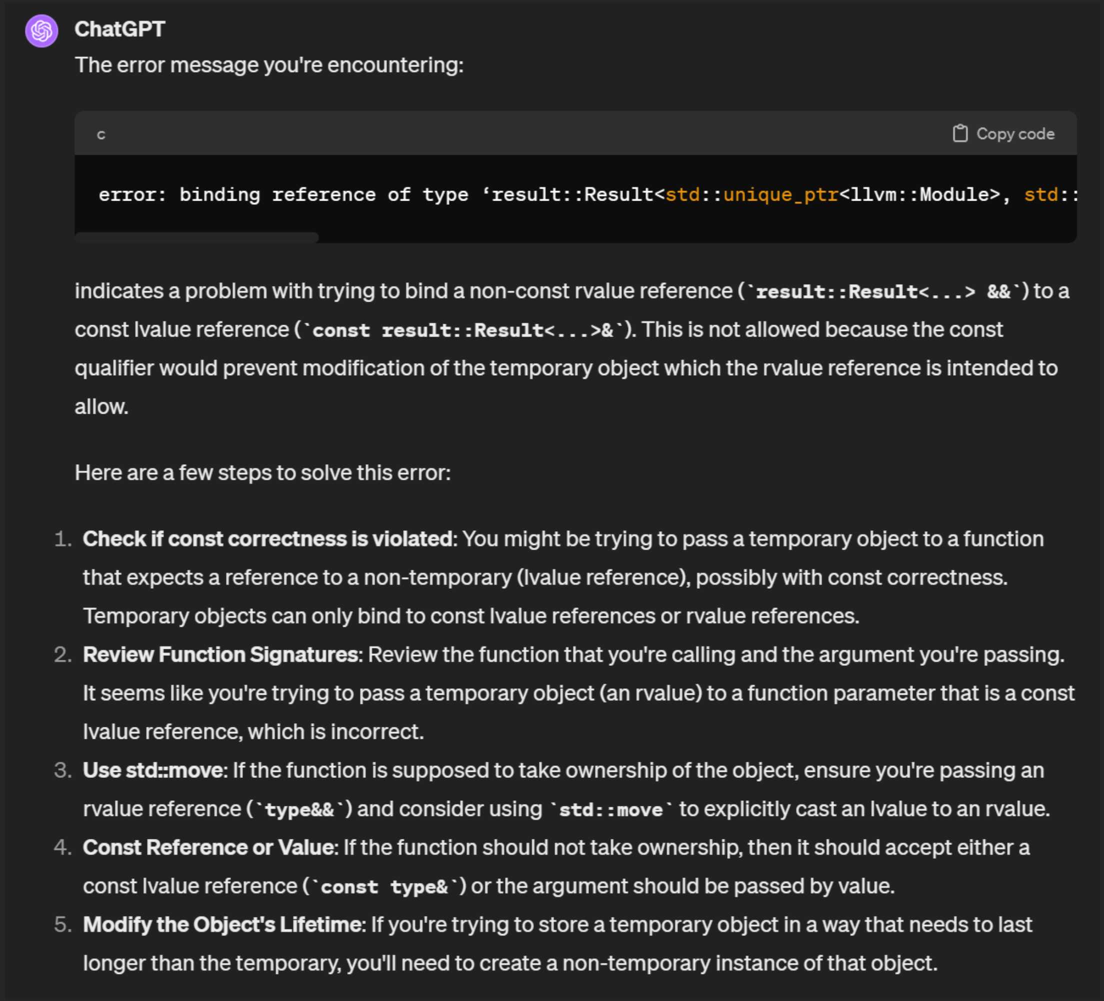
How to Ask ChatGPT



You

error: binding reference of type ‘result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >&&’ to ‘std::remove_reference<const result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >&>::type’ {aka ‘const result::Result<std::unique_ptr<llvm::Module>, std::unique_ptr<std::exception> >’} discards qualifiers

I got this error. How can I fix it?



ChatGPT

The error message you're encountering:

```c
error: binding reference of type ‘result::Result<std::unique_ptr<llvm::Module>, std:::
```

indicates a problem with trying to bind a non-const rvalue reference (`` `result::Result<...> &&` ``) to a const lvalue reference (`` `const result::Result<...>&` ``). This is not allowed because the const qualifier would prevent modification of the temporary object which the rvalue reference is intended to allow.

Here are a few steps to solve this error:

1. **Check if const correctness is violated**: You might be trying to pass a temporary object to a function that expects a reference to a non-temporary (lvalue reference), possibly with const correctness. Temporary objects can only bind to const lvalue references or rvalue references.
2. **Review Function Signatures**: Review the function that you're calling and the argument you're passing. It seems like you're trying to pass a temporary object (an rvalue) to a function parameter that is a const lvalue reference, which is incorrect.
3. **Use std::move**: If the function is supposed to take ownership of the object, ensure you're passing an rvalue reference (`` `type&&` ``) and consider using `` `std::move` `` to explicitly cast an lvalue to an rvalue.
4. **Const Reference or Value**: If the function should not take ownership, then it should accept either a const lvalue reference (`` `const type&` ``) or the argument should be passed by value.
5. **Modify the Object's Lifetime**: If you're trying to store a temporary object in a way that needs to last longer than the temporary, you'll need to create a non-temporary instance of that object.