Understanding #`ifndef` in C++

- Header Guards & Best Practices

What is `#ifndef`?

- `#ifndef` = 'if not defined'
- Used in header files to prevent multiple inclusion
- It's a preprocessor directive

```
#ifndef MYSTACK_H
#define MYSTACK_H
// Header content
#endif
```

Why is it important?

- ✅ Prevents compiler errors like:
- error: redefinition of class 'Stak'
- ✅ Avoids duplicate declarations
- ✅ Makes code scalable and modular
- ✅ Crucial in large projects with many includes

How does it work?

- 1. First time included:
- - MYSTACK_H is *not* defined → file is included
- 2. Next time:
- - MYSTACK_H *is* defined → file is skipped

- 🧠 Ensures class/function is declared only once

Best Practice Format

- `#ifndef FILENAME_H`
- `#define FILENAME_H`

- `// class or function declarations`

- `#endif // FILENAME_H`

- ✅ Use uppercase and underscores
- ✅ Match macro name to filename

Bonus – #pragma once

- 🔹 Alternative to #ifndef
- 🔹 Does the same thing:

- #pragma once

- ⚠️ Not standard (but widely supported)
- ✅ Cleaner, less error-prone
- ❗ Stick to #ifndef for full portability

Summary

- ✅ #ifndef protects your code from double-inclusion
- ✅ Essential for safe header file design
- ✅ Portable, reliable, and industry standard
- ✅ Use #pragma once if your compiler supports it (disputed advice)