**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ PATH OF LEARNING \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_+\_\_**

**Preprocessor Directives:**

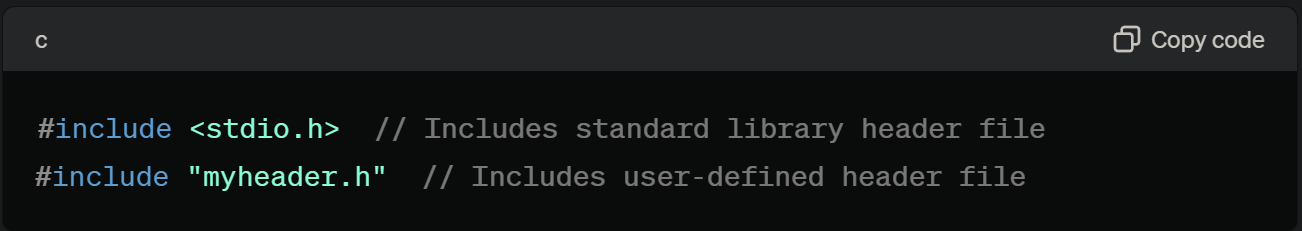
Preprocessor directives are commands in C and C++ that are executed by the preprocessor before the actual compilation of the program begins. These directives provide instructions to the compiler to preprocess the information before actual compilation starts. They typically begin with a `#` symbol and do not produce any direct executable code but rather prepare the source code for the compiler.

**Common Preprocessor Directives:**

1. `**#include**`:

- Purpose: To include the contents of another file into the current file.

- Usage:



2. `**#define**`:

- Purpose: To define macros, which are constants or expressions that can be reused throughout the code.

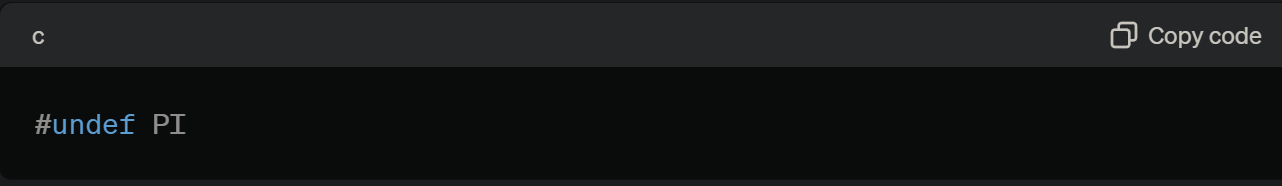
- Usage:



3. `**#undef**`:

- Purpose: To undefine a macro that was previously defined.

- Usage:



4. **Conditional Compilation Directives**:

- Purpose: To include or exclude parts of the code based on certain conditions.

- **Directives**:

- `**#ifdef**`: Checks if a macro is defined.

- `**#ifndef**`: Checks if a macro is not defined.

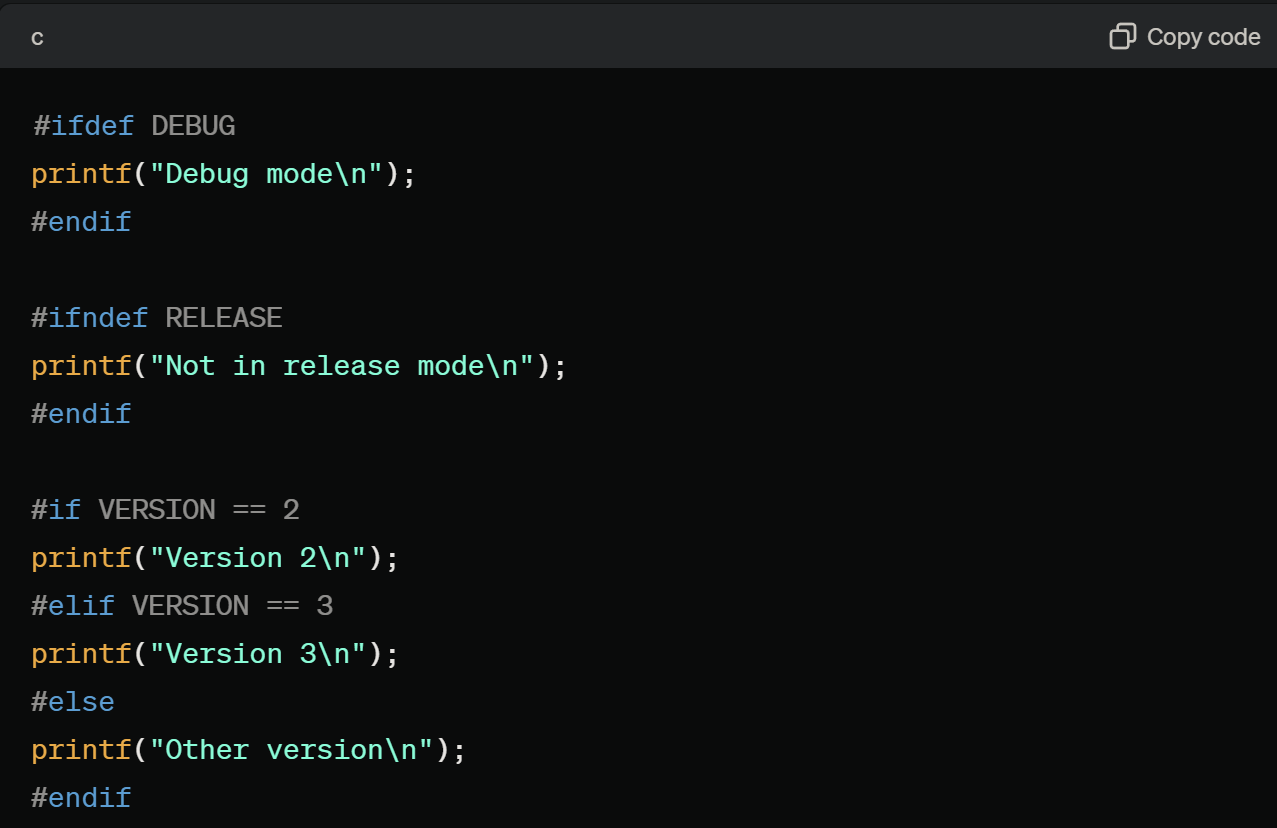
- `**#if**`: Checks if a condition is true.

- `**#else**`: Provides an alternative if the condition is false.

- `**#elif**`: Checks another condition if the previous `#if` or `#elif` was false.

- `**#endif**`: Ends the conditional directive.

- Usage:



5. `**#pragma**`:

- Purpose: To provide additional information to the compiler, which may or may not be compiler-specific.

- Usage:

