Header protection in c

Header protection in C is a mechanism used to prevent multiple inclusions of the same header file in a single translation unit, which can lead to errors such as redefinition of types, functions, and variables. This is typically achieved using **include guards** or **pragma once**.

Using Include Guards

Include guards are preprocessor directives that ensure the contents of a header file are included only once. Here's an example:

```c
#ifndef HEADER_FILE_NAME_H
#define HEADER_FILE_NAME_H

// Declarations and definitions

#endif // HEADER_FILE_NAME_H
```

- #ifndef HEADER_FILE_NAME_H checks if HEADER_FILE_NAME_H is not defined.
- #define HEADER_FILE_NAME_H defines HEADER_FILE_NAME_H.
- The actual contents of the header file (declarations and definitions) are placed between the #ifndef and #endif directives.
- #endif ends the conditional preprocessor directive.

Using #pragma once

#pragma once is a preprocessor directive that serves the same purpose as include guards but is more concise. It is not part of the C standard but is supported by most modern compilers.

Here's an example:

```c
#pragma once

// Declarations and definitions
```

Example

- *my_header.h*

- Using include guards:
- In both cases, the header file my_header.h will be included only once, preventing redefinition errors.

```c
#ifndef MY_HEADER_H
#define MY_HEADER_H

void my_function();

#endif // MY_HEADER_H
```

Using `#pragma once`:

```c
#pragma once

void my_function();
```

main.c

```c
#include "my_header.h"
#include "my_header.h" // This inclusion will be ignored due to header protection

int main() {
    my_function();
    return 0;
}
```