Preprocessor & qualifiers

C Reference

Preprocessor, const, static, enum

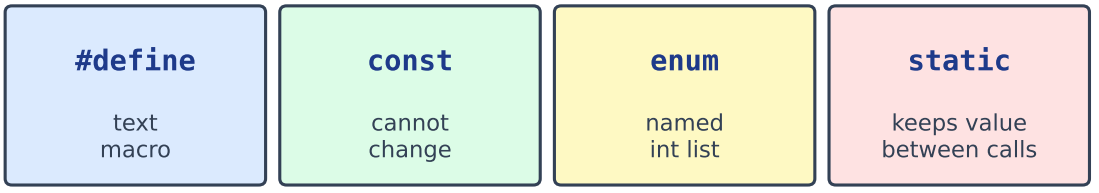
The preprocessor 预处理器 runs before compiling. `#define` makes a macro 宏 — plain text replaced everywhere. `const` makes a constant 常量 that cannot change. `static` inside a function keeps a variable's value between calls. An `enum` (enumeration 枚举) names a set of integers starting at `0`.

```c
#include <stdio.h>

#define MAX 100                      // macro: text replaced before
↪  compiling

enum Color { RED, GREEN, BLUE };     // named integers 0, 1, 2

int next_id(void) {
    static int count = 0;            // keeps its value between calls
    count++;
    return count;
}

int main(void) {
    const double PI = 3.14;          // cannot be changed
    printf("%d %.2f\n", MAX, PI);            // 100 3.14
    printf("%d %d %d\n", RED, GREEN, BLUE);  // 0 1 2
    int a = next_id();
    int b = next_id();
    printf("%d %d\n", a, b);                 // 1 2
    return 0;
}
```

Bigger programs split across files: declarations go in a header 头文件 file (`mine.h`), included with `#include "mine.h"` — quotes mean your own file, `< >` means the standard library. An include guard (`#ifndef MINE_H` / `#define MINE_H` / `#endif`) stops a header being included twice.

Common mistakes

- `#define` does plain text replacement with no type checking; wrap macro bodies and arguments in parentheses.
- A `const` value cannot be changed after it is set.
- A `static` local variable keeps its value between calls to the function.



Exam tip: preprocessor runs first; enum values start at 0 unless set

#define, const, enum, and static keep code clear