83.assert断言

1. assert 断言

`assert.h` 头文件定义了宏 `assert()` ，用于在运行时确保程序符合指定条件，如果不符合，就报错终止运行。这个宏常常被称为“断言”。

```
1 assert(p != NULL);
```

上面代码在程序运行到这一行语句时，验证变量 `p` 是否等于 `NULL` 。如果确实不等于 `NULL` ，程序继续运行，否则就会终止运行，并且给出报错信息提示。

`assert()` 宏接受一个表达式作为参数。

- 如果该表达式为真（返回值非零）， `assert()` 不会产生任何作用，程序继续运行。
- 如果该表达式为假（返回值为零）， `assert()` 就会报错，在标准错误流 `stderr` 中写入一条错误信息，显示没有通过的表达式，以及包含这个表达式的文件名和行号。

2. assert的好处

`assert()` 的使用对程序员是非常友好的，使用 `assert()` 有几个好处：

它不仅能自动标识文件和出问题的行号，还有一种无需更改代码就能开启或关闭 `assert()` 的机制。如果已经确认程序没有问题，不需要再做断言，就在 `#include <assert.h>` 语句的前面，定义一个宏 `NDEBUG` 。

```
1 #define NDEBUG
2 #include <assert.h>
```

然后，重新编译程序，编译器就会禁用文件中所有的 `assert()` 语句。如果程序又出现问题，可以移除这条 `#define NDBUG` 指令（或者把它注释掉），再次编译，这样就重新启用了 `assert()` 语句。

`assert()` 的缺点是，因为引入了额外的检查，增加了程序的运行时间。

一般我们可以在`Debug`中使用，在`Release`版本中选择禁用`assert`就行，在`VS`这样的集成开发环境中，在`Release`版本中，直接就是优化掉了。

这样在debug版本写有利于程序员排查问题，在`Release`版本不影响用户使用时程序的效率。