收录于合集
#C++小知识
16个 >

C++ 放弃具有 "nodiscard" 属性的函数的返回值？

说实话，我第一次见到C++代码里面出现这种写法感觉还是很奇特的哈。

编译环境

Visual Studio 2022 / v143 / C++17

”[[nodiscard]]

调用声明为 `nodiscard` 的函数，或调用按值返回声明为 `nodiscard` 的枚举或类的函数，则鼓励编译器发布警告。

「注意」这里是鼓励编译器发出警告，也就是编译器可以选择不发出，我这里采用msvc它是发出了警告的。

「演示 1」

```
struct [[nodiscard]] ErrorInfo
{
};
ErrorInfo process()
{
    return {};
}
ErrorInfo info;
ErrorInfo& process2()
{
    return info;
}
int main()
{
    process(); // 编译器可在舍弃 nodiscard 值时发布警告
    process2();// 并非按值返回 nodiscard 类型，无警告
   //std::ignore = process();
    return 0;
}
```

「编译输出」

```
main.cpp(15,5): warning C4834: 放弃具有 "nodiscard" 属性的函数的返回值
```

「演示 2」

标准库里面的empty方法等，也会有这个。

```
#include <vector>
int main()
{
    std::vector vec{1, 2, 3};
    vec.empty(); //  warning C4834: 放弃具有 "nodiscard" 属性的函数的返
    return 0;
}

// empty源码  可以看到 函数最前面 _NODISCARD
    _NODISCARD _CONSTEXPR20_CONTAINER bool empty() const noexcept
        auto& _My_data = _Mypair._Myval2;
        return _My_data._Myfirst == _My_data._Mylast;
    }
```

” [[fallthrough]]

指示从前一标号直落是有意的，而在发生直落时给出警告的编译器不应诊断它。

```
void f(int n)
{
    void g(), h(), i();
    switch (n) {
    case 1:
    case 2:
        g();
        [[fallthrough]];
    case 3: // 直落时不警告
        h();
    case 4: // 编译器可在发生直落时警告 - warning C4468: fallthrough: 具
        if (n < 3) {
            i();
            [[fallthrough]]; // OK
        }
        else {
            return;
        }
    case 5:
        while (false) {
            [[fallthrough]]; // 非良构：下一语句不是同一迭代的一部分
        }
        [[fallthrough]]; // OK
    case 6:
        [[fallthrough]]; // 非良构：无后继的 case 或 default 标号
    }
}
int main()
{
    return 0;
}
```

” [[mybe_unused]]

抑制针对未使用实体的警告。

```
#include <cassert>

[[maybe_unused]] void f([[maybe_unused]] bool thing1,
                        [[maybe_unused]] bool thing2)
{
    [[maybe_unused]] bool b = thing1 && thing2;
    assert(b); // Release 模式中，assert 在编译中被去掉，因而未使用 b
               // 无警告，因为它被声明为 [[maybe_unused]]
} // 未使用参数 thing1 与 thing2，无警告

int main() {}
```

参考

CPP reference

收录于合集 #C++小知识 16

上一篇
C++运算符重载模仿管道操作

下一篇
C++中switch...case绕过变量初始化

阅读原文