第40章 SFINAE

《现代C++语言核心特性解析》 谢丙堃

Substitution Failure Is Not An Error

- SFINAE全称Substitution Failure Is Not An Error，即“替换失败不是错误”。

```
template <int I> struct X {};
char foo(int);
char foo(float);
template <class T> X<sizeof(foo((T)0))> f(T)
{
    return X<sizeof(foo((T)0))>();
}

int main()
{
    f(1);
}
```

SFINAE规则详解

- 标准中规定：
  - 在直接上下文中使用模板实参替换形参后导致类型或者表达式不符合语法，那么这是替换失败；而替换后在非直接上下文中产生的副作用导致的错误则被当作编译错误。
- 这其中就包括：
  1. 处理表达式外部某些实体时发生的错误。
  2. 由于实现限制导致的错误。
  3. 由于访问违规导致的错误。
  4. 由于同一个函数的不同声明的词法顺序不同，导致替换顺序不同或者根本不发替换产生的错误。

SFINAE规则详解

- 处理表达式外部某些实体时发生的错误

```
template<class T> T foo(T& t) {
    T tt(t);
    return tt;
}
void foo(...) {}
class bar {
public:
    bar() {};
    bar(bar&&) {};
};
int main() {
    bar b;
    foo(b);
}
```

SFINAE规则详解

- 由于访问违规导致的错误

```
template<class T>
T foo(T*)
{
    return T();
}
void foo(...) {}
class bar
{
    bar() {};
};
int main()
{
    foo(static_cast<bar *>(nullptr));
}
```

SFINAE规则详解

- 由于同一个函数的不同声明的词法顺序不同，导致替换顺序不同或者根本不发替换产生的错误.

```
template <class T> struct A { using X = typename T::X; };
template <class T> typename T::X foo(typename A<T>::X);
template <class T> void foo(...) { }
template <class T> auto bar(typename A<T>::X) -> typename T::X;
template <class T> void bar(...) { }

int main()
{
    foo<int>(0);
    bar<int>(0);
}
```

SFINAE规则详解

- 非类型替换的例子:

```
template <int I> void foo(char(*)[I % 2 == 0] = 0) {
    std::cout << "I % 2 == 0" << std::endl;
}
template <int I> void foo(char(*)[I % 2 == 1] = 0) {
    std::cout << "I % 2 == 1" << std::endl;
}
int main()
{
    char a[1];
    foo<1>(&a);
    foo<2>(&a);
    foo<3>(&a);
}
```

SFINAE规则详解

- 更实际的例子:

```
class SomeObj1 {
public:
    void Dump2File() const {
        std::cout << "dump this object to file" << std::endl;
    }
};
class SomeObj2 {};
template<class T>
auto DumpObj(const T &t)->decltype(((void)t.Dump2File()), void()) {
    t.Dump2File();
}

void DumpObj(...) {
    std::cout << "the object must have a member function Dump2File" << std::endl;
}
```

感谢您的观看

欢迎关注