第15章 扩展的聚合类型

《现代C++语言核心特性解析》 谢丙堃

聚合类型的新定义

- 聚合类型需要满足的条件
  - 没有用户提供的构造函数;
  - 没有私有和受保护的非静态数据成员;
  - 没有虚函数
- 在新的扩展中，如果类存在继承关系，额外满足条件:
  - 必须是公开的基类，不能是私有或者受保护的基类;
  - 必须是非虚继承。

聚合类型的新定义

- 没有要求基类是聚合类型

```
class MyString : public std::string {};
int main()
{
    std::cout << "std::is_aggregate_v<std::string> = "
        << std::is_aggregate_v<std::string> << std::endl;
    std::cout << "std::is_aggregate_v<MyString> = "
        << std::is_aggregate_v<MyString> << std::endl;
}
```

聚合类型的初始化

- 例子

```
class MyStringWithIndex : public std::string {
public:
    int index_ = 0;
};
int main()
{
    MyStringWithIndex s{ {"hello world"}, 11 };
}
```

兼容问题

- 例子

```
class BaseData {
    int data_;
public:
    int Get() { return data_; }
protected:
    BaseData() : data_(11) {}
};

class DerivedData : public BaseData {
public:
};
DerivedData d{};
```

禁止聚合类型使用用户声明的构造函数

- 例子

```
struct X {
    X() = default;
};
struct Y {
    Y() = delete;
};

int main() {
    std::cout << std::boolalpha
        << "std::is_aggregate_v<X> : " << std::is_aggregate_v<X> << std::endl
        << "std::is_aggregate_v<Y> : " << std::is_aggregate_v<Y> << std::endl;
}
```

使用带小括号的列表初始化聚合类型对象

- 例子

```
struct X {
    int i;
    float f;
};

X x{ 11, 7.0f };
X x( 11, 7.0f );
```

感谢聆听
欢迎关注