C++17对聚合体进行了扩展,使得更多情况能够使用聚合初始化

在C++17中满足以下条件的对象被认为是聚合体:
1.是一个数组
2.是一个满足下列条件的类类型
2.1 没有用户定义的何explicit的构造函数
2.2 没有使用using声明继承的构造函数
2.3 没有private和protected的非静态数据成员
2.4 没有virtual函数
2.5 没有virtual,private,protected的基类

定义如下类

```
struct Base {
    std::string a;
    int b;
};

struct derived : Base {
    int c;
};
```

使用聚合初始化

```
   derived a{{"abc", 1}, 2};//C++17起聚合体可以拥有基类,该行为聚合初始化
   derived b{"abc", 1, 2};//OK
```

C++17以前,对于有基类的初始化需如下

```
    struct derived : Base {
        int c;
        derived(std::string a_, int b_, int c_) : Base{a_, b_}, c(c_) {}
    };

    derived c{"123", 1, 1};//OK 但是不是聚合初始化而是调用构造函数
```

Demo

```
struct Data {
        const char *p;
        double value;
    };

    struct CppData : Data {
        bool a;
    };

    CppData x{};
    CppData y{{"abc"}}; // {{"abc",0.},false}
```

```
CppData z{{}, true}; // {{nullptr,0.},true}

struct test1 {
    int a, b;
    test1(int x, int y) : a(x), b(y) {}
};

struct test2 : public test1 {
    int c;
};

test2 test{{1, 2}, 3};//从非聚合体派生出聚合体
```