条款7_1:区别()与{}创建对象

①{}的引入 struct A{A(int a){};①

A(const A&a){};②

struct B{B(int b, int c){};①

B(const B&b){};②

共存

A a=10; ①+②

A a(10); ②

A a={10}; ①+②

A a{10}; ①

A a={10}; ①

C++14前共存

1. A a=10; 问题1:仅能传递1个参数 问题2:额外进行一次拷贝

2. A a(10); 问题:被作用参数或返回值时仍会拷贝

3. A a{10}; 的优势:

{}能完美解决上述问题,但不允许缩窄转换(避免精度丢失)

可使用 static_cast<int>(x) 强制转换

大大简化了聚合类的初始化

见pdf:2

☆{}对C++最令人头痛的解析问题天生免疫(类内初始化无法使用(),被解析为函数)

②most vexing parse:

a.对象参数的创建 解析错误! b.函数声明

例1: void f(double value){

int i(int(value)); } a.新建-[int类型变量,(将value强转为int)

此时编译器只会把i认成函数,而非变量!!! 此时创建临时对象而非类型强转 b.声明-[函数 int i(int value);

C语言允许参数被括号包裹!]

例2: struct Timer{};

a. struct Timersave{ Timersave(Timer t){}};

b. Timersave time_saver(Timer());

a.创建Timersave构建函数,传入参数为Timer,构建Timersave对象

b.声明了time_saver函数,拥有匿名参数,该参数为Timer的指针(由函数退化)

③聚合类定义与不同标准的区别

定义:可通过"聚合初始化"(列表,{})直接初始化所有成员

C++14 a.所有成员均为public c.没有类内初始化 C++17取消

(但可以使用default)

b.没有定义任何构造函数 d.无基类,无虚函数

问题:初始化任务交给用户

C++17 可有基类,但必须为公有继承,且必须是非虚继承

```
class MyString: public std::string{
public: int index = 0;}
MyString s = {("Hi"), 0};
```

总是假设基类是一种在所有数据成员之前声明的特殊成员

可省略

④奇怪的array (cpp中数组的更优选择)

```
int a[3]{1,2,3};
std::array<int, 3> arr {1, 2, 3};
Point{ int arr[2];}
```

等价

Point p1[3]{{1,2},{3,4},{5,6}}; ✓

std::array<Point,3> p2{{1,2},{5,6},{8,9}}; ✗

原因：temple<typename T, size_t N>

struct Myarr{ T arr[N]; }

底层仍为原生数组

⑤如何让A支持窄式转换

C++11引入 initializer_list<> emc P84 C++P P147

注意：1.禁止隐式缩窄转换

2.优先级最高(除非不得已，否则优先使用

initializer_list<>，即使会报错)

3.空的{}，不会调用std::ini…构造，但{{}},({})

仍会调用std::init…

size为0

size为1

元素为0