|  |  |
| --- | --- |
| 1. 避免有二义性的类型转换 2. 实参匹配和相同的类型转换 3. 二义性与转换目标为内置类型的多重类型转换 4. 重载函数与转换构造函数 5. 重载函数与用户定义的类型转换 | 实参匹配和相同的类型转换  Class A{  A(const B&);  }  Class B{  Operator A() const;  }  A f(const A&);  B b;  A a = f(b)  F(b.operator A()) f(A(b))  二义性与转换目标为内置类型的多重类型转换  Struct A{  A（int= 0）;  A(double);  Operator int()const;  Operator double()const;}  Void f2(long double);  A a;  F2(a);// 二义性 int，double  Long lg;  A a2(lg);//二义性 int,double  Short s = 42;  A a3(s);// A(INT)  重载函数与转换构造函数  Class C{C(int);} class D{D(int)}  Void manip(const C&);  Void manip(const D&);  Manip(10);//二义性  Manip(C(10);//显示  重载函数与用户定义的类型转换  Struct E{E(double)}  Void manip2(const C&)  Void manip2(const E&)  Manip2(10);// 二 |
| 1. 避免有二义性的类型转换 2. 如果类中包含一个或多个类型转换，则必须确保在类类型和目标类型之间只存在唯一一种转换方式。否则存在二义性 3. 实参匹配和相同的类型转换 4. 二义性与转换目标为内置类型的多重类型转换   第一种：两个类提供相同的类型转换  A类定义一个接受B类对象的转换构造函数  B类定义了一个转换目标是A类的类型转换运算符  第二种：类定义了多个转换规则，类型转换运算符  算术运算符，这些转换涉及的类型本身可以通过其他类型转换联系到一起  第一种：  只能显式指定哪个，但是不能使用强制类型转换来解决，因为强制转换也面临二义性  第二种：  定义：类定义一组类型转换，他们的转换目标可以通过其他类型联系到一起，则二义性  标准库类型转换将决定最佳匹配到底是哪个,shourt提升int，会选int，但是double，向下int还是向上long是匹配不了。  note：除了显示地向bool类型的转换之外，尽量避免定义类型转换函数并尽可能地限制那些 “显然正确"的非显示构造函数   1. 重载函数与转换构造函数 2. 当我们调用重载的函数时，从多个类型转换中进行选择将变得更加复杂 3. 如果两个或多个类型转换都提供了同一种可行匹配，则这些类型转换一样好 4. 简单的列子：两个类都有int转换构造函数，两个重载函数都有这个类对象，调用时传入int值应该调用哪个重载函数 5. 显示的指定是哪个类对象可以解决 6. note：若需要使用构造函数或强制类型转换来改变实参的类型，意味程序设计补足 7. 重载函数与用户定义的类型转换 8. 引出：调用重载函数，若多个用户定义的类型转换都提供了可行匹配，则认为这些类型转换一样好， 9. 这一过程不会考虑任何可能出现的标准库转换的优先。   只有当重载函数只能通过一个类型转换函数得到匹配时，才考虑标准库类型优先级，像 上面提到的第二种(二义性与转换目标为内置类型的多重类型转换) | |