**类型转换**

1. **初始化和赋值进行的转换 ：**

|  |  |
| --- | --- |
| **转换** | **潜在问题** |
| 将较大的浮点类型转换为较小的浮点类型 ， 如 double 转换为float | 精度降低 ， 值可能超出目标类型的取值范围 ， 结果将不准确 |
| 将浮点数转换为整形 | 小数部分丢失 ， 原来的值可能超出目标类型的取值范围 ， 这种情况下 ， 结果将是不准确的 |
| 将较大的整形转换为较小的整形时 | 原来的值如果超出了目标类型的取值范围 ， 通常只复制右边字节 ； |

1. **以｛｝方式进行初始化时进行的转换 （C++11）**

列表初始化不允许缩窄内容 ， 即如果转换后不改变原值的准确性 ， 则允许被缩窄 ， 否则拒绝；

注意 ：

1. 这个是对应的是 c++11 的列表初始化 ， 也就是说在诸如 vector , int a[ ] = {…}等的赋值中 ， 是被允许准换的
2. **表达式中的转换：**

当一个表达式中包含两种不同的算数类型时 ， C++ 将执行两种自动转换 ， 首先 ， 一些类型在出现时便会自动转换 ； 其次 ， 有些类型在其他类型出现在表达式中时将被转换 ；

自动转换 ： 在计算表达式时 ， c++ 将 bool , char , unsigned char , signed char ,和 short 值转换为 int （**整形提升**） ；

将不同类型进行算数运算时 ， 较小的类型被转换为较大的类型

1. **强制类型转换 ：**

通用格式： （typename）value ;

typename ( value ) ;

类型转换运算符：

**dynamic\_cast :**

用于类中 ， 他解决是否可以安全将对象的地址赋给特定类型的指针问题 ， 如果可以返回指向对象的指针 ， 否则返回 NULL ；

**const\_cast :**

可以修改一个值的 const 与 volatile 状态 ；但如果是一个常量指针指向一个常量的对象则无法修改 ；

**static\_cast :**

仅当可以隐式转换时 ， 这种转换才被许可 ， 否则将

出