|  |  |
| --- | --- |
| 1. 重载、类型转换与运算符 2. 类型转换运算符 3. 定义含有类型转换运算符的类 4. 类型转换运算符可能产生意外结果 5. 显式的类型转换运算符 6. 转换为bool | 类类型转换  用户定义的类型转换  Operator type() const;  类型转换运算符  Operator int() const{return val;}  SmallInt si;  Si = 4;  Si + 3;  Int operator int() const;// 错 不能有返回值  operator int(int = 0) const;// 错 不能有参数  operator int\*() const{return 42;}// 错42不是只做  显式的类型转换运算符  Explicit Operator int()const{return val;}  SmallInt si = 3;  Si + 3;//错  Static\_cast<int>(si)+3;  Cin<<I;,早期的bool类型转换运算符，现在不可  While（std::cin>>value） |
| 1. 重载、类型转换与运算符 2. 转换构造函数和类型转换运算符共同定义了：类类型转换，也称：用户定义的类型转换 3. 转换构造函数：将其他类型转换为本类对象，定义构造函数就可以，非exclipt 4. 类型转换运算符：将本类转换成其他类型, operator type() const 5. 类型转换运算符可以面向任意类型，除了void之外，并且需要该类型作为函数的返回类型，所以数组或者函数类型不可，但可以转换成指针，或者引用 6. 类型转换运算符 7. 没有显式的返回类型，也没有形参 8. 必须定义成类的成员函数 9. 通常不应该改变待转换对象的内容 const 10. 定义含有类型转换运算符的类 11. 尽管编译器一次只能执行一次类型转换，但是能将得到后的类型转换再转换成任何其它算术类型 12. 类型转换运算符是隐式执行的，所以无法给这些函数传递实参，相应也不能定义任何形参 13. 虽然不负责指定返回类型，但实际上回返回一个对应类型的值   note:避免过度使用类型转换函数  特别是Date类型的对象和int类型的值之间不存在一对一映射关系，因此，不定义该类型转换运算符也许会更好   1. 类型转换运算符可能产生意外结果   1) 因为许多内置类型定义了bool类型转换运算符  并且bool是一种算术类型，可以将bool运用到任何需要算术类型的上下文中  Cin<<I;//即将cin转换bool，在左移位，早期版本，现在没有了   1. 显式的类型转换运算符 2. 为了防止上述问题，新引入了显式的类型转换运算符。 3. (static\_cast<int>(si1) + 3) 4. 存在一个例外，当表达式被用作条件，编译器会将显式的类型自动转换应用于它   当表达式出现在下列位置时，显式的类型转换将被隐式地执行   1. if while do 2. for 3. !|| && 4. ?: 5. 转换为bool 6. 早期：IO类型定义了向void\*转换规则 7. 现在：Io标准库通过定义一个向bool的显式类型转换实现同样目的 8. 无论什么时候在条件中使用流对象，都会使用为IO类型定义的operator bool 9. 向bool的类型转换通常用在条件部分，因此operator bool一般定义explicit   While(std::cin>>value) | |