|  |  |
| --- | --- |
| 1. 函数匹配与重载运算符 | class SmallInt {  friend SmallInt operator+(const SmallInt&, const SmallInt&);  public:  SmallInt(int = 0){}  operator int() const {  cout << "operator int() const " << endl;  return val;  }  private:  std::size\_t val;  };  SmallInt operator+(const SmallInt&, const SmallInt&) {  cout << "SmallInt operator+(const SmallInt&, const SmallInt&) " << endl;  SmallInt sl;  return sl;  }  SmallInt s1, s2;  SmallInt s3 = s1 + s2;  //int val1 = s1 + 0;// 二义性  内置还是非成员+  内置：将s1转换为int  非成员+:将0转换为smallInt |
| 1. 函数匹配与重载运算符   重载的运算符也是重载的函数。所以通用的函数匹配规则同样适用于判断在给定的表达式中用内置运算符还是重载的运算符  1.当运算符函数出现在表达式中，候选函数集的规模要比使用调用运算符调用函数时更大。  就是：a + b比 a.operator(b)与 operator(a,b)调用的函数要多  复杂的说：  1.a + b不确定是调用成员函数还是非成员函数  2.若+运算符定义为成员函数和非成员函数，这两个是不会重载的，因为调用方式不同  3.成员函数：指针，类型对象，引用来调用，非成员：只能使用函数名  所以：  a+b:可能调用： a.operator(b)  operator(a,b)  或内置运算符  而  a.operator(b) 只调用当前函数  operator(a,b)只调用当前函数  前提是需要有转换构造函数与类型转换函数就会出现多个匹配。  Note:如果我们对同一个类既提供了转换目标是算术类型的类型转换，也提供了重载的运算符，则将会遇到重载运算符与内置运算符的二义性问题。 | |