1. 函数重载

如果同意作用域内的几个函数名字相同但形参列表不同，那么我们称之为重载函数。

如：

void print(int i)

{

cout << "整数为: " << i << endl;

}

void print(double f)

{

cout << "浮点数为: " << f << endl;

}

void print(string c)

{

cout << "字符串为: " << c << endl;

}

这些函数接受的形参类型不一样，但执行的操作非常类似，当调用这些函数时，编译器会根据传递的实参类型推断想要的是那个函数。

虽然我们定义的这两个函数不相同，但是他们有共同的名字，编辑器会根据实参类型选择不同的函数。

对于重载函数来说，他们应该在形参数量或形参类型上有所不同，因为编译器以此来对函数进行选择。不允许两个函数除了返回值类型外其它所有的要素都相同。

1. 运算符重载

适用范围：C++预定义中的运算符的操作对象只局限于基本的内置数据类型，但是对于我们自定义的类型（类）是没有办法操作的。但是大多时候我们需要对我们定义的类型进行类似的运算，这个时候就需要我们对这么运算符进行重新定义，赋予其新的功能，以满足自身的需求。

重载的运算符是带有特殊名称的函数，函数名是由关键字 operator 和其后要重载的运算符符号构成的。与其他函数一样，重载运算符有一个返回类型和一个参数列表。

格式：

<返回类型说明符> operator <运算符符号>(<参数表>)

{

     <函数体>

}

举例：

Box operator+(const Box&)；

声明加法运算符用于把两个 Box 对象相加，返回最终的 Box 对象。大多数的重载运算符可被定义为普通的非成员函数或者被定义为类成员函数。如果我们定义上面的函数为类的非成员函数，那么我们需要为每次操作传递两个参数，如下所示：

如：

Box operator+(const Box&, const Box&);