Lab6

一、问题回答

区别与联系：（1）可以将运算符重载为某个类的成员函数，这样该运算符就可以自由地访问类的数据成员，也可以直接用对象进行调用，**此时的运算符重载的使用和其他类成员函数几乎没有差别**；（2）也可以将运算符重载为某个类的友元函数，这样该运算符也可以自由地访问该类的数据成员，但**此时的运算符重载相当于一个普通函数**，不能通过类的对象进行直接调用，而要通过形参方式将对象传到重载函数内。（3）对于某个类的类方法的定义没有逻辑上的限制，而对于**运算符重载则只能重载 C++允许重载的运算符，且在运算符的优先级与结合律等方面也给出了很多限制。**

例：（1）对于操作符重载主要用于使同一运算符在不同的类中进行不同的运算，重载的操作符在逻辑上应该是一致的。如对于两种不同的对象，都用“=”运算符来表示将一个对象的数据赋到另一个对象上，由于两种对象的内部数据成员可能不同，但是这里的“=”应该表示一致的赋值的含义，故需要进行重载来使“=”在不同的类中表示不同的含义。

（2）对于类方法，主要用于类内部的数据的处理。为了实现封装性，在类的外部没有权限直接访问类的成员函数，这样使得类内部的数据成员更加安全，与此同时也就需要通过类方法来对类中的数据成员进行处理。如 point 是 Point 类的一个对象，Point 类中有 Point::setpoint()成员函数来对 Point 中的数据成员初始化。这样便可以通过调用 point.setpoint()来初始化 point 的数据成员。而假如在类外部也有一个 setpoint()函数，由于该函数并非类内部的成员函数，故其调用无法影响到 point 对象。

二、代码思路（详见代码）

（1）对于大数的存储，用一个大小为 100 的 int 型数组来存储大数，此外在 BigNum 类中还有一个 int 型变量来存储大数的位数，一个 bool 型变量存储大数的正负性，一个 bool 型变量来检测大数是否越界。

（2）对于<<运算符的重载，将其声明为 BigNum 类的友元函数。接受一个文件输出流和一个 BigNum 对象 result 的引用，首先判断对象 result 是否越界，越界输出“out of range”，然后判断 result 的正负，若为负则输出一个 ‘-’，之后输出完整的大数。

（3）对于>>运算符的重载，将其声明为 BigNum 类的友元函数。接受一个键盘输出流和一个 BigNum 对象 result 的引用，通过键盘输入流接受一个字符串，然后通过 BigNum 的构造函数将字符串转化为 BigNum 对象 result。

（4）对于+、-运算符的重载，首先接受两个 BigNum 对象 X 和 Y 并检查 X 与 Y 的位数，然后让 X、Y 中的 int 型数组 m_num【100】尾部对齐。然后定义一个 BigNum 对象 Z 用于存储加和之后的对象，此时 Z 中的数组元素全为 0。从尾部位数开始，对于加法，若 X 的当前位数加上 Y 的当前位数加上 Z 的当前位数大于 10，则 Z 的更高位加 1，依次类推，最后检查总位数。对于减法，同样从尾部开始若 X 的当前位数加上 Y 的当前位数加上 Z 的当前位数小于 0，则 Z 的更高位减 1，此数当前位多加上 10 后再进行减法。然后到最高位后判断整个大数的正负，若为负则通过一次转换将其转换为绝对值进行存储并标记该对象为负。

（5）对于主函数，通过对文件内容的读取与判断，在不同的情况下进行不同的操作。详见代码及注释