## 定义基本类型的包装类

题目：

Java 语言中，为所有基本类型均提供了一个包装类，如 Integer 是 int 的包装类等等。

请您为 C++ 的基本类型 int 或 double 定义一个相应的包装类，如命名为 Integer, Double等。包装类为用户提供相应基本类型上的、语言提供的、编程时常用的操作，包括（但不限于）解析字符串、若干（双向的）类型转换操作、算术运算、关系运算等等。其中算术运算、关系运算必须采用“重载操作符”方式来定义。

定义完毕后请务必进行测试。

接口设计示例（供参考）：

下表以表示整数的类为例（假设类名为 Interger），列出部分关键操作。

|  |  |  |
| --- | --- | --- |
| 操作类别 | 建议的函数接口（原型） | 说明 |
| 成员函数 | | |
| 构造函数 | Integer( int v =0) | 参数v是该对象所表示的整数。 |
| Integer(const char\* c\_str)  Integer(const string& str) | 参数 c\_str 指向一个C语言字符串的指针，该串是一个整数的文本（十进制形式）。该函数将字符串转换为整数。如将”-123” 转换为  -123. 如下面两个语句等价：  Integer a(100); Integer a(“100”); |
| 复制操作  (成员函数) | //Copy constructor  Integer(const Integer & rhs); | 如下面语句中创建 nn 和 nnn 时就用到了这个函数。  Integer n(2);  Integer **nn(n)**，**nnn=nn**; |
| //copy assignment  Integer& operator=(const Integer& rhs); | 对象生成之后，改变其当前值。如  nn = n; // both nn, n are objects of class Integer |
| // assignment, NOT the ‘copy’ version  Integer& operator=(int rhs); | 对象生成之后，改变其当前值。  如nn = 123; // type of nn is class Integer |
| 类型转换  (成员函数) | std::string toString() const; 或者  operator std::string() const; | 将整数转换为十进制字符串并返回 |
| int intValue() const; | 返回对象所表示的int值 |
| operator int() const; | 将对象转换为其所表示的int值。  该函数和 intValue() 等价。 |
| operator double() const; | 将对象所表示的int值转换为double值 |
| operator char() const; | 将对象所表示的int值转换为 char 值 |
| 其他类型转换 |  |
|  |  |

（接下页）

（接上页，续表）

|  |  |  |
| --- | --- | --- |
| 关系运算 | int cmp(const Integer& rhs);  //成员函数 | 比较两个Integer对象所表示数值的大小关系。  若两个对象 a 和 b 满足 a<b 则返回-1；  若两个对象 a 和 b 满足 a==b 则返回0；  若两个对象 a 和 b 满足 a>b 则返回 +1。 |
| 基于函数 cmp，重载6个关系运算符。  你可以用成员函数形式，也可以用非成员函数形式重载。 | 重载举例：以非成员函数定义 < 时  bool operator<( const Integer& a, const Integer& b) { return (a.cmp(b)==-1); }  若你用成员函数重载 >= 时  bool Integer::operator>=(const Integer& b)  { return (cmp(b) >= 0 ); } |
| 字符串转换为整数  (成员函数) | static int parse(const char\* c\_str);  static int parse(const std::string& str){  return parse( str.c\_str() );  }  你还可以增加解析8进制、16进制字符串的能力。 | 一般定义为**静态**成员函数。用法举例：  int a = Integer::parse(“123”); //a=123  std::string one(“-1”);  int b = Integer::parse( one ); //b=-1 |
| 算术运算  (和基本类型相同运算的语义相同) | 重载一元 +、 -、 ++、 --。 | 也可用非成员函数实现。注意思考  返回一个临时对象、还是返回操作数自身的引用、操作数的值是否变化。  --提示：考虑基本类型运算的这些特征。 |
| 重载特殊的赋值运算： +=, -=, /=, \*=。  这些赋值运算只能用成员函数方式重载。  Integer& operator +=(const Integer& right);  Integer& operator +=(int right);  … ...  … … | 左操作数是Integer对象，其值会被修改，  右操作数是Integer 或者 int，其值不变。  返回左操作数的引用。 |
| 非成员函数部分 | | |
| 算术运算 | 重载 二元 + - \* / 运算。如  Integer operator+(const Integer& Left, const Integer& right);  Integer operator+(const Integer& Left, int right);  Integer operator+(int Left, const Integer& right); | 运算结果是一个临时对象，操作数的值保持不变。实现应确保加法、乘法满足交换律：  a + b == b + a  a \* b == b \* a |
| 输入/输出 | 重载输入操作符 >>  [参见下文 Vec4 的说明] | 必须是非成员函数. 使用示例：  Integer x(500), y;  cin >> x >> y; |
| 重载输出操作符 <<  [参见下文 Vec4 的说明] | 必须是非成员函数. 使用示例：  Integer x; x+= 1;  cout << “x=” << x << endl; |

根据上面叙述，实现开始处要求的包装类。你也可以为其他基本类型定义类似的包装类了。