|  |
| --- |
| **BG1000. The Date class (version 1)** |
| |  |  |  |  | | --- | --- | --- | --- | | Total: | 249 | Accepted: | 81 | |
|  |
|  |
| **Time Limit: 1sec    Memory Limit:256MB**  **Description**  Implement the operator <, <=, ==, !=, >, >= with the class Date  class Date  {  public:  Date(int y=0, int m=1, int d=1);  static bool leapyear(int year);  int getYear() const;  int getMonth() const;  int getDay() const;  // add any member you need here  };  You implementation should enable the usage like this:  void f()  {  Date date1, date2(2003,1,1);  Date date3 = Date(2007,2,28);  date3 = date1;  cout << "year 1996 is leap year? " << Date::leapyear(1996) << endl;  cout << "year 1200 is leap year? " << Date::leapyear(1200) << endl;  cout << "year 1300 is leap year? " << Date::leapyear(1300) << endl;  cout << "year 1999 is leap year? " << Date::leapyear(1999) << endl;  cout << "(date1==date3)? " << (date1==date3) << endl;  cout << "(date1!=date3)? " << (date1!=date3) << endl;  cout << "(date1==date2)? " << (date1==date2) << endl;  cout << "(date1!=date2)? " << (date1!=date2) << endl;  cout << "(date1<date1)? " << (date1<date1) << endl;  cout << "(date1<=date1)? " << (date1<=date1) << endl;  cout << "(date1<date2)? " << (date1<date2) << endl;  cout << "(date1<=date2)? " << (date1<=date2) << endl;    cout << "(date1>date1)? " << (date1>date1) << endl;  cout << "(date1>=date1)? " << (date1>=date1) << endl;  cout << "(date1>date2)? " << (date1>date2) << endl;  cout << "(date1>=date2)? " << (date1>=date2) << endl;  }  The output of f() should be:  year 1996 is leap year? 1  year 1200 is leap year? 1  year 1300 is leap year? 0  year 1999 is leap year? 0  (date1==date3)? 1  (date1!=date3)? 0  (date1==date2)? 0  (date1!=date2)? 1  (date1<date1)? 0  (date1<=date1)? 1  (date1<date2)? 1  (date1<=date2)? 1  (date1>date1)? 0  (date1>=date1)? 1  (date1>date2)? 0  (date1>=date2)? 0 |

|  |  |
| --- | --- |
|  | **BG1001. The Date class (version 2)** |
|  | |  |  |  |  | | --- | --- | --- | --- | | Total: | 470 | Accepted: | 82 | |
|  |  |
|  |  |
|  | **Time Limit: 1sec    Memory Limit:256MB**  **Description**  Implement the operator ++(prefix), --(prefix), ++(postfix), --(postfix), in the class Date  class Date  {  public:  Date(int y=0, int m=1, int d=1);  static bool leapyear(int year);  int getYear() const;  int getMonth() const;  int getDay() const;  friend ostream& operator<<(ostream&, const Date&);  // add any member you need here  };  You implementation should enable the usage like this:  void f()  {  Date date(2004,2,28);  cout << "date = " << date << endl;  cout << "++date = " << ++date << endl;  cout << "--date = " << --date << endl;  cout << "date++ = " << date++ << endl;  cout << "date-- = " << date-- << endl;  cout << "date = " << date << endl;  }  The output of f() should be:  date = 2004-2-28  ++date = 2004-2-29  --date = 2004-2-28  date++ = 2004-2-28  date-- = 2004-2-29  date = 2004-2-28  提交时不需要提交operator << 重载。 |

|  |
| --- |
| **BG1002. The Date class (version 3)** |
| |  |  |  |  | | --- | --- | --- | --- | | Total: | 410 | Accepted: | 49 | |
|  |
|  |
| **Time Limit: 1sec    Memory Limit:256MB**  **Description**  Implement the operator +=, -=, +, -, in the class Date  class Date  {  public:  Date(int y=0, int m=1, int d=1);  static bool leapyear(int year);  int getYear() const;  int getMonth() const;  int getDay() const;  // add any member you need here  };  You implementation should enable the usage like this:  void f()  {  Date date = d;  cout << "date = " << date << endl;  cout << "date+1 = " << date+1 << endl;  cout << "date-1 = " << date-1 << endl;  date+=366;  cout << "date = " << date << endl;  date-=365;  cout << "date = " << date << endl;  date-=-365;  cout << "date = " << date << endl;  date+=-366;  cout << "date = " << date << endl;    cout << endl;  }  The output of f() should be:  date = 2004-2-28  date+1 = 2004-2-29  date-1 = 2004-2-27  date = 2005-2-28  date = 2004-2-29  date = 2005-2-28  date = 2004-2-28  提交时不需要提交operator << 重载。 |

|  |
| --- |
| **BG1003. The String class** |
| |  |  |  |  | | --- | --- | --- | --- | | Total: | 194 | Accepted: | 64 | |
|  |
|  |
| **Time Limit: 1sec    Memory Limit:256MB**  **Description**  Implement the String class: class String { public:     String();            // x = ""     String(const char\*);        // x = "abc"      String(const String&);    // x = other\_string     String& operator=(const char \*);     String& operator=(const String&);     String operator+(String);     char& operator[](int i);     char operator[](int i) const;     int size() const;     String& operator+=(const String&);     String& operator+=(const char\*);     friend ostream& operator<<(ostream&, const String&);     friend istream& operator>>(istream&, String&);     friend bool operator==(const String& x, const char\* s);     friend bool operator==(const String& x, const String& y);     friend bool operator!=(const String& x, const char\* s);     friend bool operator!=(const String& x, const String& y); }; The String class implemented may be used like this: f() {     String x, y, s;     cout << "Please enter two strings\n";     cin >> x >> y;     cout << "x= " << x << " , y = " << y << '\n';      cout << "s = \"" << s << "\"" << endl;     s = "abc";     cout << "s = \"" << s << "\"" << endl;      cout << "\"" << x << "\" + \"" << y << "\" = " << "\"" << x+y << "\"\n";     String z = x;     if (x != z) cout << "x corrupted!\n";     x[0] = '!';     if (x == z) cout << "write failed!\n";     cout << "exit: " << x << ' ' << z << '\n';          z = s;     if (s != z) cout << "s corrupted!\n";     s[0] = '!';     if (s == z) cout << "write failed!\n";     cout << "exit: " << s << ' ' << z << '\n';     } The f() will has the same output when String replace by string in STL. |

|  |
| --- |
| **BG1004. Complex =,-,+=,-=** |
| |  |  |  |  | | --- | --- | --- | --- | | Total: | 206 | Accepted: | 85 | |
|  |
|  |
| **Time Limit: 1sec    Memory Limit:256MB**  **Description**    You are to overload the operators +,-,+=,-= for the following Complex class, which means the add/subtract operations for complex numbers.    class Complex { public:     Complex(): real(0), imag(0) {}     Complex(int a, int b): real(a), imag(b) {}     int getReal() const { return real; }     int getImag() const { return imag; } private:     int real;     int imag; };    Your submitted source code should include the implementation of class Complex and the required operators.  No main() function should be included. |

|  |  |
| --- | --- |
|  | **BG1005. Overloading ">>", "<<"** |
|  | |  |  |  |  | | --- | --- | --- | --- | | Total: | 452 | Accepted: | 74 | |
|  |  |
|  |  |
|  | **Time Limit: 1sec    Memory Limit:256MB**  **Description**    You are to overload the operators ">>", "<<", "\*" for the Complex class, which respectively mean reading an object from a stream, sending an object to a stream,  and the "multiply" operation for complex numbers.    class Complex { public:     Complex(int a=0, int b=0): real(a), imag(b) {}     friend Complex operator\*(const Complex&, const Complex&);     friend istream& operator>>(istream&, Complex&);     friend ostream& operator<<(ostream&, const Complex&); private:     int real;     int imag; };    Your submitted source code should include the implementation of the required operators, but without the Complex class.  No main() function should be included.    **Input**  The complex numbers have the form "a+bi", or "a-bi", where "a" is an integer, "b" is a non-negative integer, and there is no space inside the string. There is white space separating consecutive complex numbers. You're ensured that the input is correct.  **Output**  To make it simple, output the complex numbers in the form "a+bi", or "a-bi", where "a" is an integer, "b" is a non-nagative integer, even if a or b is zero, e.g., "0+0i", "3+0i", "0-3i". Your output should contain nothing else but the string "a+bi" or "a-bi". |