

```

1  /*
2  -----
3  Laboratoire : labo_02
4  Fichier    : date.cpp
5  Auteur(s)  : Bruno Carvalho et David Gallay
6  Date       : 8.03.2020
7
8  But        : Declare class Date, enum class Month and functions useful for it
9  Remarque(s) :
10 Compileur  : g++ 7.4.0
11 -----*/
12
13 #ifndef DATE_H
14 #define DATE_H
15
16 #include <string>
17 #include <iostream>
18
19 enum class Month {
20     JANUARY = 1,
21     FEBRUARY,
22     MARCH,
23     APRIL,
24     MAY,
25     JUNE,
26     JULY,
27     AUGUST,
28     SEPTEMBER,
29     OCTOBER,
30     NOVEMBER,
31     DECEMBER
32 };
33
34 const unsigned DEFAULT_YEAR = 1970;
35
36 class Date {
37 public:
38     /**
39      * @brief Create a Date
40      * @param day
41      * @param month
42      * @param year
43      */
44     Date(unsigned day, unsigned month, unsigned year);
45
46     /**
47      * @brief Create a Date
48      * @param day
49      * @param month
50      * @param year
51      */
52     Date(unsigned day, std::string month, unsigned year);
53
54     /**
55      * @brief Create a Date
56      * @param day
57      * @param month
58      * @param year
59      */
60     Date(unsigned day = 1, Month month = Month::JANUARY, unsigned year = DEFAULT_YEAR);
61
62
63     /**
64      * @brief Update day value
65      * @param day
66      * @return Reference on this
67      */
68     Date& setDay(unsigned day);
69
70     /**
71      * @brief Update month value
72      * @param month
73      * @return Reference on this
74      */
75     Date& setMonth(unsigned month);
76

```

```

77      /**
78       * @brief Update month value
79       * @param month
80       * @return Reference on this
81       */
82      Date& setMonth(Month month);
83
84      /**
85       * @brief Update month value
86       * @param month
87       * @return Reference on this
88       */
89      Date& setMonth(std::string month);
90
91      /**
92       * @brief Update year value
93       * @param year
94       * @return Reference on this
95       */
96      Date& setYear(unsigned year);
97
98      /**
99       * @brief get day value
100      * @return unsigned
101      */
102      unsigned getDay() const;
103
104      /**
105       * @brief get month value
106       * @return unsigned
107       */
108      unsigned getMonthNo() const;
109
110      /**
111       * @brief get month value
112       * @return string
113       */
114      std::string getMonthString() const;
115
116      /**
117       * @brief get month value
118       * @return Month enum
119       */
120      Month getMonthEnum() const;
121
122      /**
123       * @brief get year value
124       * @return unsigned
125       */
126      unsigned getYear() const;
127
128      /**
129       * @brief get a bool indicating if Date values are valid
130       * @return bool
131       */
132      bool isValid() const;
133
134
135      /**
136       * @brief Indicate if the year is a leap year
137       * @return True if the year is leap, else return false
138       */
139      static bool isLeap(unsigned year);
140
141      /**
142       * @brief Get the number of day in a month
143       * @return Number of day in the given month or unsigned max value
144       */
145      static unsigned dayInMonth(unsigned month, unsigned year);
146
147      /**
148       * @brief Convert month to string according to the define format
149       * @return Month converted into string
150       */
151      operator std::string();
152

```

```

153     /**
154      * @brief Compare two dates
155      * @param date
156      * @return True if the two dates are equal
157      */
158     bool operator==(const Date& date) const;
159
160     /**
161      * @brief Compare two dates
162      * @param date
163      * @return True if the two dates are unequal
164      */
165     bool operator!=(const Date& date) const;
166
167     /**
168      * @brief Compare two dates
169      * @param date
170      * @return True if the date tested is lower than the param date
171      */
172     bool operator<(const Date& date) const;
173
174     /**
175      * @brief Compare two dates
176      * @param date
177      * @return True if the date tested is lower or equal to the param date
178      */
179     bool operator<=(const Date& date) const;
180
181     /**
182      * @brief Compare two dates
183      * @param date
184      * @return True if the date tested is higher than the param date
185      */
186     bool operator>(const Date& date) const;
187
188     /**
189      * @brief Compare two dates
190      * @param date
191      * @return True if the date tested is higher or equal to the param date
192      */
193     bool operator>=(const Date& date) const;
194
195     /**
196      * @brief Add X day to a date
197      * @param days
198      * @return The object date modified
199      */
200     Date& operator+=(unsigned days);
201
202     /**
203      * @brief Substract X day to a date
204      * @param days
205      * @return The object date modified
206      */
207     Date& operator-=(unsigned days);
208
209     /**
210      * @brief Add X day to a date
211      * @param days
212      * @return The object date modified
213      */
214     Date& operator+=(int days);
215
216     /**
217      * @brief Substract X day to a date
218      * @param days
219      * @return The object date modified
220      */
221     Date& operator-=(int days);
222
223     /**
224      * @brief Get the number of days between two dates
225      * @param days
226      * @return days as int
227      */
228     int operator-(const Date& date) const;

```

```

229
230     /**
231      * @brief Assign an object date to an other object date
232      * @param date
233      * @return The object date modified
234      */
235     Date& operator=(const Date& date);
236
237     /**
238      * @brief Pre-incrementation of an object date
239      * @return The object date modified
240      */
241     Date& operator++();
242
243     /**
244      * @brief Post-incrementation of an object date
245      * @return The object date before incrementation
246      */
247     Date operator++(int);
248
249     /**
250      * @brief Pre-decrementation of an object date
251      * @return The object date modified
252      */
253     Date& operator--();
254
255     /**
256      * @brief Post-decrementation of an object date
257      * @return The object date before decrementation
258      */
259     Date operator--(int);
260
261     /**
262      * @brief Display date in a format DD.MM.YYYY
263      * @param os
264      * @return output stream "DD.MM.YYYY"
265      */
266     std::ostream& display(std::ostream& os = std::cout) const;
267
268     /**
269      * @brief Receive date in a format DD.MM.YYYY
270      * @param is
271      * @return input stream "DD.MM.YYYY"
272      */
273     std::istream& receive(std::istream& is = std::cin);
274
275
276 private:
277
278     /**
279      * @brief Set validity according to inner values
280      */
281     void setValidity();
282
283     /**
284      * @brief Get the number of days since a reference date,
285      *         it is used to compute the number of day between two dates
286      */
287     int get_days_since_reference_day() const;
288     static bool isDateValid(unsigned day, unsigned month, unsigned year);
289     static bool isYearValid(unsigned year);
290     static bool isMonthValid(unsigned month);
291     static bool isDayValid(unsigned day, unsigned month, unsigned year);
292
293     bool _is_valid;
294     unsigned _day;
295     unsigned _month;
296     unsigned _year;
297
298 };
299
300 std::ostream& operator<<(std::ostream& os, const Date& date);
301 std::istream& operator>>(std::istream& is, Date& date);
302
303 Date operator+(Date date, unsigned days);
304 Date operator+(unsigned days, const Date date);

```

```
305 Date operator-(Date date, unsigned days);  
306  
307 Date operator+(Date date, int days);  
308 Date operator+(int days, const Date& date);  
309 Date operator-(Date date, int days);  
310  
311  
312 #endif  
313
```

```

1  /*
2  -----
3  Laboratoire : labo_02
4  Fichier     : date.cpp
5  Auteur(s)  : Bruno Carvalho et David Gallay
6  Date       : 8.03.2020
7
8  But        : Function definition for header date.h
9  Remarque(s) :
10 Compileur  : MinGW-g++ 6.3.0 and g++ 7.4.0
11 -----*/
12
13 #include "date.h"
14 #include <sstream>
15 #include <iomanip>
16
17 const char* const MONTH_NAME[] = {
18     "UNDEFINED",
19     "JANUARY",
20     "FEBRUARY",
21     "MARCH",
22     "APRIL",
23     "MAY",
24     "JUNE",
25     "JULY",
26     "AUGUST",
27     "SEPTEMBER",
28     "OCTOBER",
29     "NOVEMBER",
30     "DECEMBER"
31 };
32
33 size_t MONTH_NAME_SIZE = sizeof(MONTH_NAME) / sizeof(const char*);
34 const unsigned REF_YEAR = 1582;
35
36 std::string monthToString(unsigned month) {
37     if(month < MONTH_NAME_SIZE)
38         return MONTH_NAME[month];
39     return MONTH_NAME[0];
40 }
41
42 std::string toString(Month month) {
43     return monthToString(unsigned(month));
44 }
45
46 unsigned convertMonth(std::string month) {
47     for(size_t index = 0; index < MONTH_NAME_SIZE; ++index) {
48         if(MONTH_NAME[index] == month) {
49             return (unsigned)index;
50         }
51     }
52     return 0;
53 }
54
55
56 Date::Date(unsigned day, unsigned month, unsigned year): _day(day), _month(month),
57 _year(year) {
58     setValidity();
59 }
60
61 Date::Date(unsigned day, std::string month, unsigned year): Date(day, convertMonth(month),
62 year) {
63 }
64
65 Date::Date(unsigned day, Month month, unsigned year): Date(day, unsigned(month), year) {
66 }
67
68
69 Date::operator std::string() {
70
71     std::stringstream stream;
72     stream << *this;
73     return stream.str();
74 }

```

```
75
76 Date& Date::setDay(unsigned day) {
77     _day = day;
78     setValidity();
79     return *this;
80 }
81
82 Date& Date::setMonth(unsigned month) {
83     _month = month;
84     setValidity();
85     return *this;
86 }
87
88 Date& Date::setMonth(Month month) {
89     return setMonth(unsigned(month));
90 }
91
92 Date& Date::setMonth(std::string month) {
93     return setMonth(convertMonth(month));
94 }
95
96 Date& Date::setYear(unsigned year) {
97     _year = year;
98     setValidity();
99     return *this;
100 }
101
102 unsigned Date::getDay() const {
103     return _day;
104 }
105
106 unsigned Date::getMonthNo() const {
107     return _month;
108 }
109
110 std::string Date::getMonthString() const {
111     return monthToString(_month);
112 }
113
114 Month Date::getMonthEnum() const {
115     return Month(_month);
116 }
117
118 unsigned Date::getYear() const {
119     return _year;
120 }
121
122 bool Date::isLeap(unsigned year) {
123     return (!(year % 4) and year % 100) or !(year % 400);
124 }
125
126 void Date::setValidity() {
127     _is_valid = isDateValid(_day, _month, _year);
128 }
129
130 bool Date::isDateValid(unsigned day, unsigned month, unsigned year) {
131     return isYearValid(year) and isMonthValid(month) and isDayValid(day, month, year);
132 }
133
134 bool Date::isYearValid(unsigned year) {
135     return year >= REF_YEAR;
136 }
137
138 bool Date::isMonthValid(unsigned month) {
139     return unsigned(Month::JANUARY) <= month and month <= unsigned(Month::DECEMBER);
140 }
141
142 bool Date::isDayValid(unsigned day, unsigned month, unsigned year) {
143     return 0 < day and day <= dayInMonth(month, year);
144 }
145
146 bool Date::isValid() const {
147     return _is_valid;
148 }
149
150 unsigned Date::dayInMonth(unsigned month, unsigned year) {
```

```

151     switch ((Month)month) {
152         case Month::JANUARY:
153         case Month::MARCH:
154         case Month::MAY:
155         case Month::JULY:
156         case Month::AUGUST:
157         case Month::OCTOBER:
158         case Month::DECEMBER:
159             return 31;
160
161         case Month::APRIL:
162         case Month::JUNE:
163         case Month::SEPTEMBER:
164         case Month::NOVEMBER:
165             return 30;
166         case Month::FEBRUARY:
167             return isLeap(year) ? 29 : 28;
168     }
169     return -1;
170 }
171
172 std::ostream& operator<<(std::ostream& os, const Date& date){
173     return date.display(os);
174 }
175
176 std::istream& operator>>(std::istream& is, Date& date){
177     return date.receive(is);
178 }
179
180 // On doit pouvoir utiliser getline
181 std::istream & Date::receive(std::istream &is) {
182     const char DELIMITER = '.';
183     char first_delimiter;
184     char second_delimiter;
185
186     is >> _day
187     >> first_delimiter
188     >> _month
189     >> second_delimiter
190     >> _year;
191
192     if(is.fail()) {
193         _is_valid = false;
194         is.clear();
195         while(is.get() != '\n');
196     } else if (first_delimiter != DELIMITER or second_delimiter != DELIMITER){
197         _is_valid = false; // Si la date est juste mais avec les mauvais delimiter, il
198                             // suffira de setDay(getDay()), // ce n'est donc pas une bonne façon de gérer le mauvais format
199     } else {
200         setValidity();
201     }
202
203     return is;
204 }
205
206
207 std::ostream & Date::display(std::ostream &os) const{
208     if(_is_valid)
209         return os << std::setfill('0') << std::setw(2) << _day << "." << _month << "." <<
210         _year;
211     return os << "invalide";
212 }
213
214 bool Date::operator==(const Date &date) const {
215     if(_year == date._year){
216         if(_month == date._month){
217             if(_day == date._day){
218                 return true;
219             }
220         }
221     }
222     return false;
223 }
224

```



```

225 bool Date::operator!=(const Date &date) const {
226     return !(*this == date);
227 }
228
229 bool Date::operator<(const Date &date) const {
230     if(_year < date._year) {
231         return true;
232     }
233     else if(_year == date._year) {
234         if(_month < date._month) {
235             return true;
236         }
237         else if(_month == date._month) {
238             if(_day < date._day) {
239                 return true;
240             }
241         }
242     }
243     return false;
244 }
245
246 bool Date::operator>(const Date &date) const {
247     return (date < *this);
248 }
249
250 bool Date::operator<=(const Date &date) const {
251     return !(*this > date);
252 }
253
254 bool Date::operator>=(const Date &date) const {
255     return !(*this < date);
256 }
257
258 Date& Date::operator+=(unsigned days) {
259     if(!isValid())
260         return *this;
261
262     while(days) {
263         unsigned nbDays = dayInMonth(_month, _year) - _day + 1;
264         if(nbDays > days) {
265             _day += days;
266             days = 0;
267         } else {
268             days -= nbDays;
269             _day = 1;
270             if(_month == unsigned(Month::DECEMBER)) {
271                 _month = 1;
272                 ++_year;
273             } else {
274                 ++_month;
275             }
276         }
277     }
278     return *this;
279 }
280
281 Date& Date::operator-=(unsigned days) {
282     if(!isValid())
283         return *this;
284
285     while(days) {
286         if(_day > days) {
287             _day -= days;
288             days = 0;
289         } else {
290             days -= _day;
291             if(_month == unsigned(Month::JANUARY)) {
292                 _month = unsigned(Month::DECEMBER);
293                 --_year;
294             } else {
295                 --_month;
296             }
297             _day = dayInMonth(_month, _year);
298         }
299     }
300 }

```

```

301         return *this;
302     }
303 }
304
305
306 Date& Date::operator+=(int days) {
307     if(days < 0)
308         return *this -= unsigned(-days);
309     return *this += unsigned(days);
310 }
311
312 Date& Date::operator-=(int days) {
313     if(days < 0)
314         return *this += unsigned(-days);
315     return *this -= unsigned(days);
316 }
317
318 int Date::get_days_since_reference_day() const {
319
320     const int DAY_PER_YEAR    = 365;
321     const int MONTH_PER_YEAR = 12;
322     const int REFERENCE_YEAR  = 1600; // Must be a leap year
323
324     int start_of_year_shifter = (14 - _month) / MONTH_PER_YEAR;
325     int number_of_months = _month + MONTH_PER_YEAR * start_of_year_shifter - 3;
326     int number_of_years = _year - REFERENCE_YEAR - start_of_year_shifter;
327     int number_of_leap_years = number_of_years / 4 - number_of_years / 100 + number_of_years /
328     400;
329
330     int days = _day + (153 * number_of_months + 2) / 5 + DAY_PER_YEAR * number_of_years +
331     number_of_leap_years + 58;
332
333     return days;
334 }
335
336 int Date::operator-(const Date& date) const {
337     return get_days_since_reference_day() - date.get_days_since_reference_day();
338 }
339
340 Date operator+(Date date, unsigned days) {
341     return date += days;
342 }
343
344 Date operator+(unsigned days, const Date& date) {
345     return date + days;
346 }
347
348 Date operator-(Date date, unsigned days) {
349     return date -= days;
350 }
351
352 Date operator+(Date date, int days) {
353     return date += days;
354 }
355
356 Date operator+(int days, const Date& date) {
357     return date + days;
358 }
359
360 Date operator-(Date date, int days) {
361     return date -= days;
362 }
363
364 Date& Date::operator++() {
365     return *this += 1;
366 }
367
368 Date Date::operator++(int) {
369     Date temp = *this;
370     ++*this;
371     return temp;
372 }
373
374 Date& Date::operator--() {
375     return *this -= 1;

```

```
375 }
376
377 Date Date::operator--(int) {
378     Date temp = *this;
379     --*this;
380     return temp;
381 }
382
383 Date& Date::operator=(const Date &date) {
384     _day    = date._day;
385     _month  = date._month;
386     _year   = date._year;
387
388     /*
389      * safer than copying _is_valid:
390      * e.g: Suppose that a class inherits from Date and overrides
391      * functions which leads to change the behaviour of _is_valid.
392
393      * To ensure the integrity of the object, we use setValidity.
394      */
395     setValidity();
396     return *this;
397 }
398
```

```

1  /*
2  -----
3  Laboratory   : labo_02
4  File        : labo_02_Carvalho_bruno_gallay_david.cpp
5  Author(s)   : Bruno Carvalho et David Gallay
6  Date        : 8.03.2020
7
8  Purpose     : Prove the good working of classes defined in others files.
9  Remark(s)   :
10               There is the github repository:
11               https://github.com/dgheig/Ba2-labo02
12
13  Compiler    : g++ 7.4.0
14  -----*/
15  #include <iostream>
16  #include <cstdlib>
17  #include "src/date.h"
18
19  using namespace std;
20
21  #define WAIT_ENTER while(cin.get()!='\n')
22
23  int main() {
24      //TEST INPUT STREAM //
25      {
26          Date date;
27      }
28      cout << "TEST '=' OPERATOR" << endl;
29      {
30          Date date1(12,1,1990);
31          cout << date1 << endl;
32          Date date2 = date1;
33          cout << date2 << endl;
34      }
35      cout << "TEST '<' OPERATOR" << endl;
36      {
37          Date date1(12,1,1990);
38          Date date2(13, 3, 2000);
39          cout << date1 << "<" << date2 << " : ";
40          cout << boolalpha << (date1 < date2) << endl;
41          cout << date2 << "<" << date1 << " : ";
42          cout << boolalpha << (date2 < date1) << endl;
43      }
44      cout << "TEST '<=' OPERATOR" << endl;
45      {
46          Date date1(14,3,2002);
47          Date date2(1,5,1980);
48          cout << date1 << "<=" << date2 << " : ";
49          cout << boolalpha << (date1 <= date2) << endl;
50          cout << date2 << "<=" << date1 << " : ";
51          cout << boolalpha << (date2 <= date1) << endl;
52      }
53      cout << "TEST '>' OPERATOR" << endl;
54      {
55          Date date1(12,1,1990);
56          Date date2(13, 3, 2000);
57          cout << date1 << ">" << date2 << " : ";
58          cout << boolalpha << (date1 > date2) << endl;
59          cout << date2 << ">" << date1 << " : ";
60          cout << boolalpha << (date2 > date1) << endl;
61      }
62      cout << "TEST '>=' OPERATOR" << endl;
63      {
64          Date date1(14,3,2002);
65          Date date2(1,5,1980);
66          cout << date1 << ">=" << date2 << " : ";
67          cout << boolalpha << (date1 >= date2) << endl;
68          cout << date2 << ">=" << date1 << " : ";
69          cout << boolalpha << (date2 >= date1) << endl;
70      }
71      cout << "TEST '==' OPERATOR" << endl;
72      {
73          Date date1(14,5,2000);
74          Date date2(14,5,2000);
75          Date date3(15,5,2000);
76          cout << date1 << "==" << date2 << " : ";

```

```
77         cout << boolalpha << (date1 == date2) << endl;
78         cout << date1 << "==" << date3 << " : ";
79         cout << boolalpha << (date1 == date3) << endl;
80     }
81     cout << "TEST '!=' OPERATOR" << endl;
82     {
83         Date date1(14,5,2000);
84         Date date2(14,5,2000);
85         Date date3(15,5,2000);
86         cout << date1 << "!=" << date2 << " : ";
87         cout << boolalpha << (date1 != date2) << endl;
88         cout << date1 << "!=" << date3 << " : ";
89         cout << boolalpha << (date1 != date3) << endl;
90     }
91     cout << "TEST '+' OPERATOR" << endl;
92     {
93         Date date1;
94         cout << date1 << "+" << "7" << " = ";
95         cout << (date1 + 7) << endl;
96     }
97     cout << "TEST '-' OPERATOR" << endl;
98     {
99         Date date1;
100        cout << date1 << "-" << "22" << " = ";
101        cout << (date1 - 22) << endl;
102    }
103    cout << "TEST '++' OPERATOR" << endl;
104    {
105        Date date1(15,1,1997);
106        cout << "Post-increment " << date1++ << endl;
107        cout << "After Post-increment " << date1 << endl;
108        cout << "Pre-increment " << ++date1 << endl;
109    }
110    cout << "TEST '--' OPERATOR" << endl;
111    {
112        Date date1(15,1,1997);
113        cout << "Post-decrement " << date1-- << endl;
114        cout << "After Post-decrement " << date1 << endl;
115        cout << "Pre-decrement " << --date1 << endl;
116    }
117    cout << "TEST 'string()' CAST OPERATOR" << endl;
118    {
119        Date date(23,8,2007);
120        cout << string(date) << endl;
121    }
122
123
124    cout << "Please, press <ENTER> to end the program" << endl;
125    WAIT_ENTER;
126    return EXIT_SUCCESS;
127 }
128
```

```

1  /*
2  -----
3  Laboratoire : labo_02
4  Fichier      : testAccessors.cpp
5  Auteur(s)    : Bruno Carvalho et David Gallay
6  Date         : 24.02.2020
7
8  But          : Example of test file for getters and setters
9  Remarque(s)  :
10 Compileur    : g++ 7.4.0
11 -----*/
12
13 #include <iostream>
14 #include "../src/date.h"
15
16 using namespace std;
17
18 int exit_value = EXIT_SUCCESS;
19
20
21 void checkValidity(const Date& date, bool expected) {
22
23     if(date.isValid() != expected) {
24         exit_value = EXIT_FAILURE;
25         cerr << "Date validity is wrong:\n" << boolalpha
26              << date << '\n'
27              << "Expected: " << expected << '\n'
28              << "Got: " << date.isValid()
29              << endl;
30     } else {
31         cout << "OK" << endl;
32     }
33 }
34
35 void check(const Date& date, const Date& expected) {
36
37     if(date != expected) {
38         exit_value = EXIT_FAILURE;
39         cerr << "Date value is wrong:\n"
40              << "Expected: " << expected << '\n'
41              << "Got: " << date
42              << endl;
43     } else {
44         cout << "OK" << endl;
45     }
46 }
47
48 void check(unsigned value, unsigned expected) {
49
50     if(value != expected) {
51         exit_value = EXIT_FAILURE;
52         cerr << "Date value is wrong:\n"
53              << "Expected: " << expected << '\n'
54              << "Got: " << value
55              << endl;
56     } else {
57         cout << "OK" << endl;
58     }
59 }
60
61 int main() {
62
63     Date a(31, 1, 2020);
64     a.setMonth(Month::AUGUST);
65     check(a, Date(31, Month::AUGUST, 2020));
66
67     a.setDay(34);
68     checkValidity(a, false);
69
70     a.setDay(30).setMonth(2);
71     checkValidity(a, false);
72
73
74     a.setDay(29);
75     checkValidity(a, true);
76     check(a, Date(29, "FEBRUARY", 2020));

```

```
77
78     a.setYear(2019);
79     checkValidity(a, false);
80
81     check(a.getDay(), 29);
82     check(a.getMonthNo(), 2);
83     check(a.getYear(), 2019);
84
85     if(exit_value == EXIT_SUCCESS)
86         cout << "Check were successful" << endl;
87     return exit_value;
88 }
89
```

```

1  /*
2  -----
3  Laboratoire : labo_02
4  Fichier      : testComparisonOperators.cpp
5  Auteur(s)    : Bruno Carvalho et David Gallay
6  Date         : 24.02.2020
7
8  But          : Example of test file for comparison operator
9  Remarque(s)  :
10 Compileur    : g++ 7.4.0
11 -----*/
12
13 #include <iostream>
14 #include "../src/date.h"
15
16 using namespace std;
17
18 int exit_value = EXIT_SUCCESS;
19
20
21 void check(bool result, bool expected) {
22
23     if(result != expected) {
24         exit_value = EXIT_FAILURE;
25         cerr << "Comparison is wrong:\n" << boolalpha
26              << "Expected: " << expected << '\n'
27              << "Got: " << result
28              << endl;
29     } else {
30         cout << "OK" << endl;
31     }
32 }
33
34 int main() {
35     Date date1(12,1,1990);
36     Date date2(13, 3, 2000);
37     Date date3(14,3,2002);
38     Date date4(1,5,1980);
39
40
41     // operator<
42
43     cout << date1 << " < " << date2 << endl;
44     check(date1 < date2, true);
45     cout << endl;
46
47
48     // operator<=
49
50     cout << date3 << " <= " << date4 << endl;
51     check(date3 <= date4, false);
52     cout << endl;
53
54
55     // operator>
56
57     cout << date1 << " > " << date2 << endl;
58     check(date1 > date2, false);
59     cout << endl;
60
61
62     // operator>=
63
64     cout << date3 << " >= " << date4 << endl;
65     check(date3 >= date4, true);
66     cout << endl;
67
68
69     // operator==
70
71     cout << date3 << " == " << date4 << endl;
72     check(date3 == date4, false);
73     cout << endl;
74
75     cout << date4 << " == " << date4 << endl;
76     check(date4 == date4, true);

```



```
77     cout << endl;
78
79
80     // operator!=
81     cout << date3 << " != " << date4 << endl;
82     check(date3 != date4, true);
83     cout << endl;
84
85     cout << date4 << " != " << date4 << endl;
86     check(date4 != date4, false);
87     cout << endl;
88
89
90
91     if(exit_value == EXIT_SUCCESS)
92         cout << "Check were successful" << endl;
93     return exit_value;
94 }
95
```

```

1  /*
2  -----
3  Laboratoire : labo_02
4  Fichier      : testDiffDate.cpp
5  Auteur(s)    : Bruno Carvalho et David Gallay
6  Date         : 24.02.2020
7
8  But          : Example of test file for increments, decrements
9  Remarque(s) :
10 Compileur    : g++ 7.4.0
11 -----*/
12
13 #include <iostream>
14 #include "../src/date.h"
15
16 using namespace std;
17
18 int exit_value = EXIT_SUCCESS;
19
20 void check(int value, int expected) {
21     if(value != expected) {
22         exit_value = EXIT_FAILURE;
23         cerr << "Difference of day is wrong:\n"
24              << "Expected: " << expected << '\n'
25              << "Got: "      << value
26              << endl;
27     } else {
28         cout << "OK" << endl;
29     }
30 }
31
32 void check(const Date& date, const Date& expected) {
33     if(date != expected) {
34         exit_value = EXIT_FAILURE;
35         cerr << "Date value is wrong:\n"
36              << "Expected: " << expected << '\n'
37              << "Got: "      << date
38              << endl;
39     } else {
40         cout << "OK" << endl;
41     }
42 }
43
44 int main() {
45     Date date1(12,1,1990);
46     Date date2(13, 3, 2000);
47     Date date3(14,3,2002);
48     Date date4(1,5,1980);
49
50     Date date5(31, 1, 2020);
51     Date date6(1, 1, 2020);
52
53     cout << date5 << " - " << date6 << endl;
54     check(date5 - date6, 30);
55     cout << endl;
56
57     cout << date1 + 3713 << " - " << date2 << endl;
58     check(date1 + 3713 , date2);
59     cout << endl;
60
61     cout << date1 - 3543 << " - " << date4 << endl;
62     check(date1 - 3543 , date4);
63     cout << endl;
64
65     cout << date1 << "++" << endl;
66     check(date1++ , Date(12,1,1990));
67     check(date1 , Date(13,1,1990));
68     cout << endl;
69
70     cout << "++" << date1 << endl;
71     check(++date1 , Date(14,1,1990));
72     check(date1 , Date(14,1,1990));
73     cout << endl;

```

```
77
78     cout << date1 << "--" << endl;
79     check(date1--, Date(14,1,1990));
80     check(date1 , Date(13,1,1990));
81     cout << endl;
82
83     cout << "--" << date1 << endl;
84     check(--date1 , Date(12,1,1990));
85     check(date1 , Date(12,1,1990));
86     cout << endl;
87
88
89     if(exit_value == EXIT_SUCCESS)
90         cout << "Check were successful" << endl;
91     return exit_value;
92 }
93
```