```
/*
-----------------------------------------------------------------------------------
Laboratoire : labo_02
Fichier     : date.cpp
Auteur(s)   : Bruno Carvalho et David Gallay
Date        : 8.03.2020

But         : Declare class Date, enum class Month and functions useful for it
Remarque(s) :
Compilateur : g++ 7.4.0
-----------------------------------------------------------------------------------*/

#ifndef DATE_H
#define DATE_H

#include <string>
#include <iostream>

enum class Month {
    JANUARY = 1,
    FEBRUARY,
    MARCH,
    APRIL,
    MAY,
    JUNE,
    JULY,
    AUGUST,
    SEPTEMBER,
    OCTOBER,
    NOVEMBER,
    DECEMBER
};

const unsigned DEFAULT_YEAR = 1970;

class Date {
    public:
        /**
         * @brief Create a Date
         * @param day
         * @param month
         * @param year
         */
        Date(unsigned day, unsigned month, unsigned year);

        /**
         * @brief Create a Date
         * @param day
         * @param month
         * @param year
         */
        Date(unsigned day, std::string month, unsigned year);

        /**
         * @brief Create a Date
         * @param day
         * @param month
         * @param year
         */
        Date(unsigned day = 1, Month month = Month::JANUARY, unsigned year = DEFAULT_YEAR);


        /**
         * @brief Update day value
         * @param day
         * @return Reference on this
         */
        Date& setDay(unsigned day);

        /**
         * @brief Update month value
         * @param month
         * @return Reference on this
         */
        Date& setMonth(unsigned month);

        /**
         * @brief Update month value
         * @param month
         * @return Reference on this
         */
        Date& setMonth(Month month);

        /**
         * @brief Update month value
         * @param month
         * @return Reference on this
         */
        Date& setMonth(std::string month);

        /**
         * @brief Update year value
         * @param year
         * @return Reference on this
         */
        Date& setYear(unsigned year);

        /**
         * @brief get day value
         * @return unsigned
         */
        unsigned getDay() const;

        /**
         * @brief get month value
         * @return unsigned
         */
        unsigned getMonthNo() const;

        /**
         * @brief get month value
         * @return string
         */
        std::string getMonthString() const;

        /**
         * @brief get month value
         * @return Month enum
         */
        Month getMonthEnum() const;

        /**
         * @brief get year value
         * @return unsigned
         */
        unsigned getYear() const;

        /**
         * @brief get a bool indicating if Date values are valid
         * @return bool
         */
        bool isValid() const;


        /**
         * @brief Indicate if the year is a leap year
         * @return True if the year is leap, else return false
         */
        static bool isLeap(unsigned year);

        /**
         * @brief Get the number of day in a month
         * @return Number of day in the given month or unsigned max value
         */
        static unsigned dayInMonth(unsigned month, unsigned year);

        /**
         * @brief Convert month to string according to the define format
         * @return Month converted into string
         */
        operator std::string();

        /**
         * @brief Compare two dates
         * @param date
         * @return True if the two dates are equal
         */
        bool operator==(const Date& date) const;

        /**
         * @brief Compare two dates
         * @param date
         * @return True if the two dates are inequal
         */
        bool operator!=(const Date& date) const;

        /**
         * @brief Compare two dates
         * @param date
         * @return True if the date tested is lower than the param date
         */
        bool operator<(const Date& date) const;

        /**
         * @brief Compare two dates
         * @param date
         * @return True if the date tested is lower or equal to the param date
         */
        bool operator<=(const Date& date) const;

        /**
         * @brief Compare two dates
         * @param date
         * @return True if the date tested is higher than the param date
         */
        bool operator>(const Date& date) const;

        /**
         * @brief Compare two dates
         * @param date
         * @return True if the date tested is higher or equal to the param date
         */
        bool operator>=(const Date& date) const;

        /**
         * @brief Add X day to a date
         * @param days
         * @return The object date modified
         */
        Date& operator+=(unsigned days);

        /**
         * @brief Substract X day to a date
         * @param days
         * @return The object date modified
         */
        Date& operator-=(unsigned days);

        /**
         * @brief Add X day to a date
         * @param days
         * @return The object date modified
         */
        Date& operator+=(int days);

        /**
         * @brief Substract X day to a date
         * @param days
         * @return The object date modified
         */
        Date& operator-=(int days);

        /**
         * @brief Get the number of days between two dates
         * @param days
         * @return days as int
         */
        int operator-(const Date& date) const;
```

```

            /**
             * @brief Assign an object date to an other object date
             * @param date
             * @return The object date modified
             */
            Date& operator=(const Date& date);

            /**
             * @brief Pre-incrementation of an object date
             * @return The object date modified
             */
            Date& operator++();

            /**
             * @brief Post-incrementation of an object date
             * @return The object date before incrementation
             */
            Date operator++(int);

            /**
             * @brief Pre-decrementation of an object date
             * @return The object date modified
             */
            Date& operator--();

            /**
             * @brief Post-decrementation of an object date
             * @return The object date before decrementation
             */
            Date operator--(int);

            /**
             * @brief Display date in a format DD.MM.YYYY
             * @param os
             * @return output stream "DD.MM.YYYY"
             */
            std::ostream& display(std::ostream& os = std::cout) const;

            /**
             * @brief Receive date in a format DD.MM.YYYY
             * @param is
             * @return  input stream "DD.MM.YYYY"
             */
            std::istream& receive(std::istream& is = std::cin);


        private:

            /**
             * @brief Set validity according to inner values
             */
            void setValidity();

            /**
             * @brief Get the number of days since a reference date,
             *        it is used to compute the number of day between two dates
             */
            int get_days_since_reference_day() const;
            static bool isDateValid(unsigned day, unsigned month, unsigned year);
            static bool isYearValid(unsigned year);
            static bool isMonthValid(unsigned month);
            static bool isDayValid(unsigned day, unsigned month, unsigned year);

            bool _is_valid;
            unsigned _day;
            unsigned _month;
            unsigned _year;

    };

    std::ostream& operator<<(std::ostream& os, const Date& date);
    std::istream& operator>>(std::istream& is, Date& date);

    Date operator+(Date date, unsigned days);
    Date operator+(unsigned days, const Date date);
```

```
Date operator-(Date date, unsigned days);

Date operator+(Date date, int days);
Date operator+(int days, const Date& date);
Date operator-(Date date, int days);


#endif

```

```cpp
/*
-----------------------------------------------------------------------------------
Laboratoire : labo_02
Fichier     : date.cpp
Auteur(s)   : Bruno Carvalho et David Gallay
Date        : 8.03.2020

But         : Function definition for header date.h
Remarque(s) :
Compilateur : MinGW-g++ 6.3.0 and g++ 7.4.0
-----------------------------------------------------------------------------------*/

#include "date.h"
#include <sstream>
#include <iomanip>

const char* const MONTH_NAME[] = {
    "UNDEFINED",
    "JANUARY",
    "FEBRUARY",
    "MARCH",
    "APRIL",
    "MAY",
    "JUNE",
    "JULY",
    "AUGUST",
    "SEPTEMBER",
    "OCTOBER",
    "NOVEMBER",
    "DECEMBER"
};

size_t MONTH_NAME_SIZE = sizeof(MONTH_NAME) / sizeof(const char*);
const unsigned REF_YEAR = 1582;

std::string monthToString(unsigned month) {
    if(month < MONTH_NAME_SIZE)
        return MONTH_NAME[month];
    return MONTH_NAME[0];
}

std::string toString(Month month) {
    return monthToString(unsigned(month));
}

unsigned convertMonth(std::string month) {
    for(size_t index = 0; index < MONTH_NAME_SIZE; ++index) {
        if(MONTH_NAME[index] == month) {
            return (unsigned)index;
        }
    }
    return 0;
}


Date::Date(unsigned day, unsigned month, unsigned year): _day(day), _month(month),
_year(year) {
    setValidity();
}

Date::Date(unsigned day, std::string month, unsigned year): Date(day, convertMonth(month),
year) {

}

Date::Date(unsigned day, Month month, unsigned year): Date(day, unsigned(month), year) {

}


Date::operator std::string() {

    std::stringstream stream;
    stream << *this;
    return stream.str();
}
```

```cpp

Date& Date::setDay(unsigned day) {
    _day = day;
    setValidity();
    return *this;
}

Date& Date::setMonth(unsigned month) {
    _month = month;
    setValidity();
    return *this;
}

Date& Date::setMonth(Month month) {
    return setMonth(unsigned(month));
}

Date& Date::setMonth(std::string month) {
    return setMonth(convertMonth(month));
}

Date& Date::setYear(unsigned year) {
    _year = year;
    setValidity();
    return *this;
}

unsigned Date::getDay() const {
    return _day;
}

unsigned Date::getMonthNo() const {
    return _month;
}

std::string Date::getMonthString() const {
    return monthToString(_month);
}

Month Date::getMonthEnum() const {
    return Month(_month);
}

unsigned Date::getYear() const {
    return _year;
}

bool Date::isLeap(unsigned year) {
    return (!(year % 4) and year % 100) or !(year % 400);
}

void Date::setValidity() {
    _is_valid = isDateValid(_day, _month, _year);
}

bool Date::isDateValid(unsigned day, unsigned month, unsigned year) {
    return isYearValid(year) and isMonthValid(month) and isDayValid(day, month, year);
}

bool Date::isYearValid(unsigned year) {
    return year >= REF_YEAR;
}

bool Date::isMonthValid(unsigned month) {
    return unsigned(Month::JANUARY) <= month and month <= unsigned(Month::DECEMBER);
}

bool Date::isDayValid(unsigned day, unsigned month, unsigned year) {
    return 0 < day and day <= dayInMonth(month, year);
}

bool Date::isValid() const {
    return _is_valid;
}

unsigned Date::dayInMonth(unsigned month, unsigned year) {
    switch ((Month)month) {
        case Month::JANUARY:
        case Month::MARCH:
        case Month::MAY:
        case Month::JULY:
        case Month::AUGUST:
        case Month::OCTOBER:
        case Month::DECEMBER:
            return 31;

        case Month::APRIL:
        case Month::JUNE:
        case Month::SEPTEMBER:
        case Month::NOVEMBER:
            return 30;
        case Month::FEBRUARY:
            return isLeap(year) ? 29 : 28;
    }
    return -1;
}

std::ostream& operator<<(std::ostream& os, const Date& date){
    return date.display(os);
}

std::istream& operator>>(std::istream& is,  Date& date){
    return date.receive(is);
}

// On doit pouvoir utiliser getline
std::istream & Date::receive(std::istream &is)  {
    const char DELIMITER = '.';
    char first_delimiter;
    char second_delimiter;

    is >> _day
       >> first_delimiter
       >> _month
       >> second_delimiter
       >> _year;

    if(is.fail()) {
        _is_valid = false;
        is.clear();
        while(is.get() != '\n');
    } else if (first_delimiter != DELIMITER or second_delimiter != DELIMITER){
        _is_valid = false;  // Si la date est juste mais avec les mauvais delimiter, il
        suffira de setDay(getDay()),
                            // ce n'est donc pas une bonne façon de gérer le mauvais format
    } else {
        setValidity();
    }

    return is;
}


std::ostream & Date::display(std::ostream &os) const{
    if(_is_valid)
        return os << std::setfill('0') << std::setw(2) << _day  << "." << _month << "." <<
        _year;
    return os << "invalide";

}

bool Date::operator==(const Date &date) const {
    if(_year == date._year){
        if(_month == date._month){
            if(_day == date._day){
                return true;
            }
        }
    }
    return false;
}

```

```cpp
bool Date::operator!=(const Date &date) const {
    return !(*this == date);
}

bool Date::operator<(const Date &date) const {
    if(_year < date._year) {
        return true;
    }
    else if(_year == date._year) {
        if(_month < date._month) {
            return true;
        }
        else if(_month == date._month) {
            if(_day < date._day) {
                return true;
            }
        }

    }
    return false;
}

bool Date::operator>(const Date &date) const {
    return (date < *this);
}

bool Date::operator<=(const Date &date) const {
    return !(*this > date);
}

bool Date::operator>=(const Date &date) const {
    return !(*this < date);
}

Date& Date::operator+=(unsigned days) {
    if(!isValid())
        return *this;

    while(days) {
        unsigned nbDays = dayInMonth(_month, _year) - _day + 1;
        if(nbDays > days) {
            _day += days;
            days = 0;
        } else {
            days -= nbDays;
            _day = 1;
            if(_month == unsigned(Month::DECEMBER)) {
                _month = 1;
                ++_year;
            } else {
                ++_month;
            }
        }
    }
    return *this;
}

Date& Date::operator-=(unsigned days) {
    if(!isValid())
        return *this;

    while (days) {
        if (_day > days) {
            _day -= days;
            days = 0;
        } else {
            days -= _day;
            if (_month == unsigned(Month::JANUARY)) {
                _month = unsigned(Month::DECEMBER);
                --_year;
            } else {
                --_month;
            }
            _day = dayInMonth(_month, _year);
        }
    }
    return *this;

}


Date& Date::operator+=(int days) {
    if(days < 0)
        return *this -= unsigned(-days);
    return *this += unsigned(days);
}

Date& Date::operator-=(int days) {
    if(days < 0)
        return *this += unsigned(-days);
    return *this -= unsigned(days);
}

int Date::get_days_since_reference_day() const {

   const int DAY_PER_YEAR   = 365;
   const int MONTH_PER_YEAR = 12;
   const int REFERENCE_YEAR = 1600; // Must be a leap year

   int start_of_year_shifter = (14 - _month) / MONTH_PER_YEAR;
   int number_of_months = _month + MONTH_PER_YEAR * start_of_year_shifter - 3;
   int number_of_years = _year - REFERENCE_YEAR - start_of_year_shifter;
   int number_of_leap_years = number_of_years / 4 - number_of_years / 100 + number_of_years /
   400;

   int days = _day + (153 * number_of_months + 2) / 5 + DAY_PER_YEAR * number_of_years +
   number_of_leap_years + 58;

   return days;

}

int Date::operator-(const Date& date) const {
    return get_days_since_reference_day() - date.get_days_since_reference_day();
}

Date operator+(Date date, unsigned days) {
    return date += days;
}

Date operator+(unsigned days, const Date& date) {
    return date + days;
}

Date operator-(Date date, unsigned days) {
    return date -= days;
}

Date operator+(Date date, int days) {
    return date += days;
}

Date operator+(int days, const Date& date) {
    return date + days;
}

Date operator-(Date date, int days) {
    return date -= days;
}

Date& Date::operator++() {
    return *this += 1;
}

Date Date::operator++(int) {
    Date temp = *this;
    ++*this;
    return temp;
}

Date& Date::operator--() {
    return *this -= 1;
```

```cpp
}

Date Date::operator--(int) {
    Date temp = *this;
    --*this;
    return temp;
}

Date& Date::operator=(const Date &date) {
    _day   = date._day;
    _month = date._month;
    _year  = date._year;

    /*
        safer than copying _is_valid:
        e.g: Suppose that a class inherits from Date and overrides
        functions which leads to change the behaviour of _is_valid.

        To ensure the integrity of the object, we use setValidity.
    */
    setValidity();
    return *this;
}

```

```cpp
/*
-----------------------------------------------------------------------------------
Laboratory  : labo_02
File        : labo_02_Carvalho_bruno_gallay_david.cpp
Author(s)   : Bruno Carvalho et David Gallay
Date        : 8.03.2020

Purpose     : Prove the good working of classes defined in others files.
Remark(s)   :
                There is the github repository:
                https://github.com/dgheig/Ba2-labo02

Compiler    : g++ 7.4.0
-----------------------------------------------------------------------------------*/
#include <iostream>
#include <cstdlib>
#include "src/date.h"

using namespace std;

#define WAIT_ENTER while(cin.get()!='\n')

int main() {
    //TEST INPUT STREAM //
    {
        Date date;
    }
    cout << "TEST '=' OPERATOR" << endl;
    {
        Date date1(12,1,1990);
        cout << date1 << endl;
        Date date2 = date1;
        cout << date2 << endl;
    }
    cout << "TEST '<' OPERATOR" << endl;
    {
        Date date1(12,1,1990);
        Date date2(13, 3, 2000);
        cout << date1 << "<" << date2 << " : ";
        cout << boolalpha << (date1 < date2) << endl;
        cout << date2 << "<" << date1 << " : ";
        cout << boolalpha << (date2 < date1) << endl;
    }
    cout << "TEST '<=' OPERATOR" << endl;
    {
        Date date1(14,3,2002);
        Date date2(1,5,1980);
        cout << date1 << "<=" << date2 << " : ";
        cout << boolalpha << (date1 <= date2) << endl;
        cout << date2 << "<=" << date1 << " : ";
        cout << boolalpha << (date2 <= date1) << endl;
    }
    cout << "TEST '>' OPERATOR" << endl;
    {
        Date date1(12,1,1990);
        Date date2(13, 3, 2000);
        cout << date1 << ">" << date2 << " : ";
        cout << boolalpha << (date1 > date2) << endl;
        cout << date2 << ">" << date1 << " : ";
        cout << boolalpha << (date2 > date1) << endl;
    }
    cout << "TEST '>=' OPERATOR" << endl;
    {
        Date date1(14,3,2002);
        Date date2(1,5,1980);
        cout << date1 << ">=" << date2 << " : ";
        cout << boolalpha << (date1 >= date2) << endl;
        cout << date2 << ">=" << date1 << " : ";
        cout << boolalpha << (date2 >= date1) << endl;
    }
    cout << "TEST '==' OPERATOR" << endl;
    {
        Date date1(14,5,2000);
        Date date2(14,5,2000);
        Date date3(15,5,2000);
        cout << date1 << "==" << date2 << " : ";
        cout << boolalpha << (date1 == date2) << endl;
        cout << date1 << "==" << date3 << " : ";
        cout << boolalpha << (date1 == date3) << endl;
    }
    cout << "TEST '!=' OPERATOR" << endl;
    {
        Date date1(14,5,2000);
        Date date2(14,5,2000);
        Date date3(15,5,2000);
        cout << date1 << "!=" << date2 << " : ";
        cout << boolalpha << (date1 != date2) << endl;
        cout << date1 << "!=" << date3 << " : ";
        cout << boolalpha << (date1 != date3) << endl;
    }
    cout << "TEST '+' OPERATOR" << endl;
    {
        Date date1;
        cout << date1 << "+" << "7" << " = ";
        cout << (date1 + 7) << endl;
    }
    cout << "TEST '-' OPERATOR" << endl;
    {
        Date date1;
        cout << date1 << "-" << "22" << " = ";
        cout << (date1 - 22) << endl;
    }
    cout << "TEST '++' OPERATOR" << endl;
    {
        Date date1(15,1,1997);
        cout << "Post-increment " << date1++ << endl;
        cout << "After Post-increment " << date1 << endl;
        cout << "Pre-increment " << ++date1 << endl;
    }
    cout << "TEST '--' OPERATOR" << endl;
    {
        Date date1(15,1,1997);
        cout << "Post-decrement " << date1-- << endl;
        cout << "After Post-decrement " << date1 << endl;
        cout << "Pre-decrement " << --date1 << endl;
    }
    cout << "TEST 'string()' CAST OPERATOR" << endl;
    {
        Date date(23,8,2007);
        cout << string(date) << endl;
    }


    cout << "Please, press <ENTER> to end the program" << endl;
    WAIT_ENTER;
    return EXIT_SUCCESS;
}

```

```
/*
-----------------------------------------------------------------------------------
Laboratoire : labo_02
Fichier     : testAccessors.cpp
Auteur(s)   : Bruno Carvalho et David Gallay
Date        : 24.02.2020

But         : Example of test file for getters and setters
Remarque(s) :
Compilateur : g++ 7.4.0
-----------------------------------------------------------------------------------*/

#include <iostream>
#include "../src/date.h"

using namespace std;

int exit_value = EXIT_SUCCESS;


void checkValidity(const Date& date, bool expected) {

    if(date.isValid() != expected) {
        exit_value = EXIT_FAILURE;
        cerr << "Date validity is wrong:\n" << boolalpha
             << date << '\n'
             << "Expected: " << expected << '\n'
             << "Got: "      << date.isValid()
             << endl;
    } else {
        cout << "OK" << endl;
    }
}

void check(const Date& date, const Date& expected) {

    if(date != expected) {
        exit_value = EXIT_FAILURE;
        cerr << "Date value is wrong:\n"
             << "Expected: " << expected << '\n'
             << "Got: "      << date
             << endl;
    } else {
        cout << "OK" << endl;
    }
}

void check(unsigned value, unsigned expected) {

    if(value != expected) {
        exit_value = EXIT_FAILURE;
        cerr << "Date value is wrong:\n"
             << "Expected: " << expected << '\n'
             << "Got: "      << value
             << endl;
    } else {
        cout << "OK" << endl;
    }
}

int main() {

    Date a(31, 1, 2020);
    a.setMonth(Month::AUGUST);
    check(a, Date(31, Month::AUGUST, 2020));

    a.setDay(34);
    checkValidity(a, false);

    a.setDay(30).setMonth(2);
    checkValidity(a, false);


    a.setDay(29);
    checkValidity(a, true);
    check(a, Date(29, "FEBRUARY", 2020));

    a.setYear(2019);
    checkValidity(a, false);

    check(a.getDay(), 29);
    check(a.getMonthNo(), 2);
    check(a.getYear(), 2019);

    if(exit_value == EXIT_SUCCESS)
        cout << "Check were successful" << endl;
    return exit_value;
}

```

```cpp
/*
-----------------------------------------------------------------------------------
Laboratoire : labo_02
Fichier     : testComparisonOperators.cpp
Auteur(s)   : Bruno Carvalho et David Gallay
Date        : 24.02.2020

But         : Example of test file for comparison operator
Remarque(s) :
Compilateur : g++ 7.4.0
-----------------------------------------------------------------------------------*/

#include <iostream>
#include "../src/date.h"

using namespace std;

int exit_value = EXIT_SUCCESS;


void check(bool result, bool expected) {

    if(result != expected) {
        exit_value = EXIT_FAILURE;
        cerr << "Comparison is wrong:\n" << boolalpha
             << "Expected: " << expected << '\n'
             << "Got: "      << result
             << endl;
    } else {
        cout << "OK" << endl;
    }
}

int main() {
    Date date1(12,1,1990);
    Date date2(13, 3, 2000);
    Date date3(14,3,2002);
    Date date4(1,5,1980);


    // operator<

    cout << date1 << " < " << date2 << endl;
    check(date1 < date2, true);
    cout << endl;


    // operator<=

    cout << date3 << " <= " << date4 << endl;
    check(date3 <= date4, false);
    cout << endl;


    // operator>

    cout << date1 << " > " << date2 << endl;
    check(date1 > date2, false);
    cout << endl;


    // operator>=

    cout << date3 << " >= " << date4 << endl;
    check(date3 >= date4, true);
    cout << endl;


    // operator==

    cout << date3 << " == " << date4 << endl;
    check(date3 == date4, false);
    cout << endl;

    cout << date4 << " == " << date4 << endl;
    check(date4 == date4, true);
    cout << endl;


    // operator!=
    cout << date3 << " != " << date4 << endl;
    check(date3 != date4, true);
    cout << endl;

    cout << date4 << " != " << date4 << endl;
    check(date4 != date4, false);
    cout << endl;



    if(exit_value == EXIT_SUCCESS)
        cout << "Check were successful" << endl;
    return exit_value;
}

```

```cpp
/*
-----------------------------------------------------------------------------------
Laboratoire : labo_02
Fichier     : testDiffDate.cpp
Auteur(s)   : Bruno Carvalho et David Gallay
Date        : 24.02.2020

But         : Example of test file for increments, decrements
Remarque(s) :
Compilateur : g++ 7.4.0
-----------------------------------------------------------------------------------*/

#include <iostream>
#include "../src/date.h"

using namespace std;

int exit_value = EXIT_SUCCESS;

void check(int value, int expected) {

    if(value != expected) {
        exit_value = EXIT_FAILURE;
        cerr << "Difference of day is wrong:\n"
             << "Expected: " << expected << '\n'
             << "Got: "      << value
             << endl;
    } else {
        cout << "OK" << endl;
    }
}

void check(const Date& date, const Date& expected) {

    if(date != expected) {
        exit_value = EXIT_FAILURE;
        cerr << "Date value is wrong:\n"
             << "Expected: " << expected << '\n'
             << "Got: "      << date
             << endl;
    } else {
        cout << "OK" << endl;
    }
}

int main() {
    Date date1(12,1,1990);
    Date date2(13, 3, 2000);
    Date date3(14,3,2002);
    Date date4(1,5,1980);

    Date date5(31, 1, 2020);
    Date date6(1, 1, 2020);

    cout << date5 << " - " << date6 << endl;
    check(date5 - date6, 30);
    cout << endl;


    cout << date1 + 3713 << " - " << date2 << endl;
    check(date1 + 3713 , date2);
    cout << endl;

    cout << date1 - 3543 << " - " << date4 << endl;
    check(date1 - 3543 , date4);
    cout << endl;

    cout << date1 << "++" << endl;
    check(date1++ , Date(12,1,1990));
    check(date1 , Date(13,1,1990));
    cout << endl;

    cout << "++" << date1 << endl;
    check(++date1 , Date(14,1,1990));
    check(date1 , Date(14,1,1990));
    cout << endl;

    cout << date1 << "--" << endl;
    check(date1-- , Date(14,1,1990));
    check(date1  , Date(13,1,1990));
    cout << endl;

    cout << "--" << date1 << endl;
    check(--date1  , Date(12,1,1990));
    check(date1  , Date(12,1,1990));
    cout << endl;


    if(exit_value == EXIT_SUCCESS)
        cout << "Check were successful" << endl;
    return exit_value;
}

```