CECS 342-07: Principles of Programming Languages
Program 2 – Overloading Operators in C++
Due: Sept 28, 2023

Create a C++ class called Date. It should have the following operations:

- Date( ) – default constructor. This will set the date to Jan 1, 1970 (Unix Epoch time)
- Date(int M, int D, int Y) – overloaded constructor. This will set the date to the values passed in through the parameter list represented by Month, Day and Year. If this is NOT a valid date, the date is set to the default date.
- Date(int J) – overloaded constructor – create a date using the Julian date
- ~Date( ) – destructor. Be sure to de-allocate any memory that was allocated in the constructor.
- int getMonth( ) – return the month in integer form
- int getDay( ) – return the day of the month
- int getYear( ) – return the year
- string getMonthName( ) – return the name of the month – 3 letters only: Jan, Feb, etc
- string getDayName() – returns week day name – Monday, Tuesday, etc

Add the necessary class methods (functions) to support the following:

- Date D1(10,27,2010); // overloaded constructor
- Date D2(D1); // copy constructor
- Date D3 = D2; // also copy constructor – initialize D3 to be copy of D2
- D1 = D2; // assignment operator
- D1 += 5; // add 5 days to D1, result is stored in D1
- D1 -= 7; // subtract 7 days from D1, result is stored in D1
- D3 = D2 + 5; // add 5 days to D2, assign result to D3
- D3 = 5 + D2; // add 5 days to D2, assign result to D3
- D3 = D2 – 4; // subtract 4 days from D2, assign result to D3
- int x = D5 – D4; // days between D5 and D4. Can be negative or positive
- cout << Date::count(); // a static method that returns the number of Date objects that currently exist in the program
- cout << D1.count() // same as above
- cout << D1; // will print "10/27/2010"
- D1++; // increment D1 by one day – postfix style
- ++D1; // also increment D1 by one day – prefix style
- D1--; // decrement D1 by one day – postfix style
- --D1; // decrement D1 by one day – prefix style
- cout << D1.julian(); // print the Julian integer represented by D1
- Comparison Operators
  - if (D1 == D2)
  - if (D1 < D2)
  - if (D1 > D2)

Additional requirements:

You MUST create a pointer in the private data members that points to an integer array. When you create a Date object (using any constructor) you must dynamically allocate an array of three integers – the Month, Day and Year. Similar details must happen during the assignment operator.

Write a driver program that tests each operation. I will provide a test program to you before the due date.

Learning Objectives

- Create class using separate header/cpp files
- Operator Overloading
- Friend functions
- Deep vs shallow copy – we require deep on this assignment because of dynamic memory in constructors
- Big 3 – if a class defines any of these, it should define them all
    - Copy constructor
    - Assignment operator
    - destructor
- Static data members to keep track of how many instances currently exist
- Static class member functions
- Pointers / memory leaks
- Exposure to Fortran code

Converting Between Julian Dates and Gregorian Calendar Dates

The Julian date (JD) is a continuous count of days from 1 January 4713 BC (= -4712 January 1), Greenwich mean noon (= 12h UT). For example, AD 1978 January 1, 0h UT is JD 2443509.5 and AD 1978 July 21, 15h UT, is JD 2443711.125.

Conversion of Gregorian calendar date to Julian date for years AD 1801-2099 can be carried out with the following formula:

JD = 367K - <(7(K+<(M+9)/12>))/4> + <(275M)/9> + I + 1721013.5 + UT/24 - 0.5sign(100K+M-190002.5) + 0.5

where K is the year (1801 <= K <= 2099), M is the month (1 <= M <= 12), I is the day of the month (1 <= I <= 31), and UT is the universal time in hours ("<=" means "less than or equal to"). The last two terms in the formula add up to zero for all dates after 1900 February 28, so these two terms can be omitted for subsequent dates. This formula makes use of the sign and truncation functions described below:

The **sign** function serves to extract the algebraic sign from a number.
Examples: sign(247) = 1; sign(-6.28) = -1.

The **truncation** function < > extracts the integral part of a number.
Examples: <17.835> = 17; <-3.14> = -3.

Example: Compute the JD corresponding to 1877 August 11, 7h30m UT.
Substituting K = 1877, M = 8, I = 11 and UT = 7.5,
JD = 688859 - 3286 + 244 + 11 + 1721013.5 + 0.3125 + 0.5 + 0.5
= 2406842.8125

The formula given above was taken from the U.S. Naval Observatory's no-longer- published *Almanac for Computers* for year 1990.

Also see our Julian date converter data service.

Fliegel and van Flandern (1968) published compact computer algorithms for converting between Julian dates and Gregorian calendar dates. Their algorithms were presented in the Fortran programming language, and take advantage of the truncation feature of integer arithmetic. The following Fortran code modules are based on these algorithms. In this code, YEAR is the full representation of the year, such as 1970, 2000, etc. (not a two-digit abbreviation); MONTH is the month, a number from 1 to 12; DAY is the day of the month, a number in the range 1-31; and JD is the the Julian date at Greenwich noon on the specified YEAR, MONTH, and DAY.

Conversion from a Gregorian calendar date to a Julian date. Valid for any Gregorian calendar date producing a Julian date greater than zero:

```
      INTEGER FUNCTION JD (YEAR,MONTH,DAY)
C
C---COMPUTES THE JULIAN DATE (JD) GIVEN A GREGORIAN CALENDAR
C   DATE (YEAR,MONTH,DAY).
C
      INTEGER YEAR,MONTH,DAY,I,J,K
C
      I= YEAR
      J= MONTH
      K= DAY
C
      JD= K-32075+1461*(I+4800+(J-14)/12)/4+367*(J-2-(J-14)/12*12)
     2    /12-3*((I+4900+(J-14)/12)/100)/4
C
      RETURN
      END
```

Conversion from a Julian date to a Gregorian calendar date.

```
      SUBROUTINE GDATE (JD, YEAR,MONTH,DAY)
C
C---COMPUTES THE GREGORIAN CALENDAR DATE (YEAR,MONTH,DAY)
C   GIVEN THE JULIAN DATE (JD).
C
      INTEGER JD,YEAR,MONTH,DAY,I,J,K
C
      L= JD+68569
      N= 4*L/146097
      L= L-(146097*N+3)/4
      I= 4000*(L+1)/1461001
      L= L-1461*I/4+31
      J= 80*L/2447
      K= L-2447*J/80
      L= J/11
      J= J+2-12*L
      I= 100*(N-49)+I+L
C
      YEAR= I
      MONTH= J
      DAY= K
C
      RETURN
      END
```

Example: YEAR = 1970, MONTH = 1, DAY = 1, JD = 2440588.

Reference: Fliegel, H. F. and van Flandern, T. C. (1968). Communications of the ACM, Vol. 11, No. 10 (October, 1968).