

```
/*
  TITLE: dateclass.h
  AUTHOR: V. Ree
  CREATION DATE: Sunday 27 April 2003
  REVISION: 0
  LAST REVISION DATE:
  REVISION HISTORY:
    0 Initial issue.
  DESCRIPTION:
    Definition of date class.
*/

#include <iostream.h>
#include <time.h>
#include <string.h>
using std::cout;
using std::endl;

// Definition of Date class.

class Date {

  // Member Functions
  public:

    // Constructor is overloaded.
    Date();
    Date( int, int ); // ddd yyyy
    Date( int, int, int ); // mm/dd/yy or mm/dd/yyyy
    Date( char *, int, int ); // monthname dd yyyy

    // Set functions
    void setMonth( int );
    void setDay( int );
    void setYear( int );

    // Print functions
    void printDate_DDD_YYYY( void ) const;
    void printDate_MM_DD_YY( void ) const;
    void printDate_MonthName_DD_YYYY( void ) const;

    // Conversion functions

    // Determines Month and Day from ddd and Year.
    void ddd_to_mmddyyyy( int );

    // Converts Date to a ddd yyyy format
    int date_to_dddyyyy( void ) const;

    // Translates month, written as text, into month number and applies to Month.
    void monthname_to_monthnumber( const char * const );

    // Other utility functions

    // Returns the month name based on month.
    const char *monthName( void ) const;

    // Determines if year is a leap year.
    bool leapYear( void ) const;

    // Returns the number of days in month.
    int daysOfMonth( void ) const;

    // Returns name of month based on month index.
    const char *monthList( int ) const;

    // Returns number of days in month.

```

```
    int days( int ) const;

    // Member data
private:
    int day;
    int month;
    int year;
};

// CONSTRUCTORS

// Date constructor that uses day of year and year
Date::Date( int ddd, int yyyy )
{
    setYear( yyyy );
    ddd_to_mmddyyyy( ddd ); // Determines month and day from ddd and Year.
}

// Date constructor that uses month, day and year
Date::Date( int mm, int dd, int yy )
{
    setYear( yy + 1900 ); // Assumes 19xx is default for a two-digit year.
    setMonth( mm );
    setDay( dd );
}

// Date constructor that uses month name, day and year
Date::Date( char *mPtr, int dd, int yyyy )
{
    setYear( yyyy );

    monthname_to_monthnumber( mPtr ); // Translates month, written as text,
                                     // into month number and applies to month.
    setDay( dd );
}

// Date constructor that uses functions from time.h
// Will set Date to "today".
Date::Date()
{
    struct tm *ptr; // pointer of type struct tm
                  // which holds calendar time components

    time_t t = time( 0 ); // determine the current calendar time
                        // which is assigned to timePtr

    ptr = localtime( &t ); // convert the current calendar time
                          // pointed to by timePtr into
                          // broken down time and assign it to ptr

    day = ptr->tm_mday; // broken down day of month

    month = 1 + ptr->tm_mon; // broken down month since January

    year = ptr->tm_year + 1900; // broken down year since 1900
}

// SET FUNCTIONS

// Set the day
void Date::setDay( int d )
{
    day = d >= 1 && d <= daysOfMonth() ? d : 1; // Default day of month is 1.
}
}
```

```
// Set the month
void Date::setMonth( int m )
{
    month = m >= 1 && m <= 12 ? m : 1; // Default month is January.
}

// Set the year
void Date::setYear( int y )
{
    /* year = y >= 1900 && y <= 1999 ? y : 1900; // Default year is 1900. */
    year = y >= 1900 && y <= 2099 ? y : 1900; // Default year is 1900.
}

// PRINT FUNCTIONS

// Print Date in the form: ddd yyyy
void Date::printDate_DDD_YYYY( void ) const
{
    cout << date_to_dddyyyy() << ' ' << year << '\n';
}

// Print Date in the form: mm/dd/yyyy
void Date::printDate_MM_DD_YY( void ) const
{
    cout << month << '/' << day << '/' << year << '\n';
}

// Print Date in the form: monthname dd, yyyy
void Date::printDate_MonthName_DD_YYYY( void ) const
{
    cout << monthName() << ' ' << day << ", " << year << '\n';
}

// Return the Date object month name.
const char *Date::monthName( void ) const
{
    int month_index;

    month_index = month - 1;
    return monthList( month_index );
}

// Return the number of days in the Date object month.
int Date::daysOfMonth( void ) const
{
    return leapYear() && month == 2 ? 29 : days( month );
}

// Test for a leap year
bool Date::leapYear( void ) const
{
    // A year is a leap year if:
    // year is divisible by 400 OR
    // year is divisible by 4 AND NOT divisible by 100.
    // "x % y == 0" is the same as "x is divisible by y".
    // "x % y != 0" is the same as "x is not divisible by y".
    if ( ( year % 400 == 0 ) || ( ( year % 4 == 0 ) && ( year % 100 != 0 ) ) )
    {
        return true; // year is a leap year
    }
    else
    {
        return false; // year is a leap year
    }
}
```

```
// Converts ddd to mm / dd / yyyy format.
void Date::ddd_to_mmddyyyy( int ddd )
{
    int dayTotal = 0;
    int dayTotalTemp = 0;

    if ( ( ddd < 1 ) || ( ddd > 366 ) ) // check for invalid day
    {
        ddd = 1; // if invalid, set ddd=1 (January 1).
    }

    setMonth( 1 ); // month = 1 = January
    while ( dayTotalTemp < ddd )
    {
        dayTotalTemp = dayTotal + daysOfMonth();

        if ( dayTotalTemp < ddd )
        {
            dayTotal = dayTotal + daysOfMonth();
            setMonth( month + 1 );
        }
        else
        {
            setDay( daysOfMonth() - ( dayTotalTemp - ddd ) );
        }
    }
}

// Converts Date to a ddd yyyy format
int Date::date_to_dddyyyy( void ) const
{
    int ddd = 0;
    int m; // month number, index for loop.

    for ( m = 1; m < month; ++m )
    {
        ddd = ddd + days( m );
    }

    // Add leap day if necessary.
    if ( leapYear() && month >= 2 )
    {
        ddd = ddd + 1;
    }

    ddd = ddd + day;

    return ddd;
}

// Converts month name to month number
void Date::monthname_to_monthnumber( const char * const mPtr )
{
    int month_index;
    bool flag = false; // flag to indicate search complete.

    // Search: Cycle through each month until a match is found.
    for ( month_index = 0; month_index < 12; ++month_index )
    {
        if ( !strcmp( mPtr, monthList( month_index ) ) )
        {
            setMonth( month_index + 1 );
            flag = true; // set flag; search was successful.
            break; // stop checking for month
        }
    }
}
```

```
    }
}

// Search failed (invalid month name).
// Set month to January.
if ( !flag )
{
    setMonth( 1 );
}

// Return the name of the month based on month index.
// month index: 0=January, 1=February, ..., 11=December
const char *Date::monthList( int mm ) const
{
    char *months[] = { "January", "February", "March",
                      "April", "May", "June",
                      "July", "August", "September",
                      "October", "November", "December" };
    return months[ mm ];
}

// Return the number days in the month based on month index.
// month index: 0=January, 1=February, ..., 11=December
int Date::days( int m ) const
{
    const int monthDays[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    return monthDays[ m - 1 ]; // Remember, monthDays[0] = 31 = days in January
}
```