Algorithms Level 4

PROGRAMMING ADVICES

LEARN THE RIGHT WAY

26+ Years
of Experience

Mohammed Abu-Hadhoud

MBA, PMOC, PgMP®, PMP®, PMI-RMP®, CM, ITILF, MCPD, MCSD

ProgrammingAdvices.com

```cpp
#pragma warning(disable : 4996)

#include <iostream>
using namespace std;

struct stDate
{
    short Year;
    short Month;
    short Day;

};

bool isLeapYear(short Year)
{
   return (Year % 4 == 0 && Year % 100 != 0) || (Year % 400 == 0);
}

bool IsDate1BeforeDate2(stDate Date1, stDate Date2)
{
    return  (Date1.Year < Date2.Year) ? true : ((Date1.Year ==
Date2.Year) ? (Date1.Month < Date2.Month ? true : (Date1.Month ==
Date2.Month ? Date1.Day < Date2.Day : false)) : false);
}

short NumberOfDaysInAMonth(short Month, short Year)
{

    if (Month < 1 || Month>12)
        return  0;

    int days[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
    return (Month == 2) ? (isLeapYear(Year) ? 29 : 28) :
days[Month - 1];

}

bool IsLastDayInMonth(stDate Date)
{
    return (Date.Day == NumberOfDaysInAMonth(Date.Month,
Date.Year));

}

bool IsLastMonthInYear(short Month)
{
    return (Month == 12);
}
```

```
stDate IncreaseDateByOneDay(stDate Date)
{
    if (IsLastDayInMonth(Date))
    {
        if (IsLastMonthInYear(Date.Month))
        {
            Date.Month = 1;
            Date.Day = 1;
            Date.Year++;
        }
        else
        {
            Date.Day = 1;
            Date.Month++;
        }
    }
    else
    {
        Date.Day++;
    }

    return Date;
}

int GetDifferenceInDays(stDate Date1, stDate Date2, bool
IncludeEndDay = false)
{

    int Days = 0;
    while (IsDate1BeforeDate2(Date1, Date2))
    {
        Days++;
        Date1 = IncreaseDateByOneDay(Date1);
    }

    return IncludeEndDay ? ++Days : Days;
}
```

```
short DayOfWeekOrder(short Day, short Month, short Year)
{
    short a, y, m;
    a = (14 - Month) / 12;
    y = Year - a;
    m = Month + (12 * a) - 2;
    // Gregorian:
    //0:sun, 1:Mon, 2:Tue...etc
    return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m)
/ 12)) % 7;
}

short DayOfWeekOrder(stDate Date)
{
    return DayOfWeekOrder(Date.Day, Date.Month, Date.Year);

}

string DayShortName(short DayOfWeekOrder)
{
    string arrDayNames[] = {
"Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

    return arrDayNames[DayOfWeekOrder];

}

short IsEndOfWeek(stDate Date)
{
    return  DayOfWeekOrder(Date) == 6;
}

bool IsWeekEnd(stDate Date)
{
    //Weekends are Fri and Sat
    short DayIndex = DayOfWeekOrder(Date);
    return  (DayIndex == 5 || DayIndex == 6);
}
```

```
bool IsBusinessDay(stDate Date)
{
    //Weekends are Sun,Mon,Tue,Wed and Thur

   /* short DayIndex = DayOfWeekOrder(Date);
    return  (DayIndex >= 5 && DayIndex <= 4);
   */

   //shorter method is to invert the IsWeekEnd: this will save
updating code.
    return !IsWeekEnd(Date);

}

short DaysUntilTheEndOfWeek(stDate Date)
{
    return 6 - DayOfWeekOrder(Date);

}

short DaysUntilTheEndOfMonth(stDate Date1)
{

    stDate EndOfMontDate;
    EndOfMontDate.Day = NumberOfDaysInAMonth(Date1.Month,
Date1.Year);
    EndOfMontDate.Month = Date1.Month;
    EndOfMontDate.Year = Date1.Year;

    return GetDifferenceInDays(Date1, EndOfMontDate, true);

}

short DaysUntilTheEndOfYear(stDate Date1)
{

    stDate EndOfYearDate;
    EndOfYearDate.Day = 31;
    EndOfYearDate.Month = 12;
    EndOfYearDate.Year = Date1.Year;

    return GetDifferenceInDays(Date1, EndOfYearDate, true);

}
```

```
short ReadDay()
{
    short Day;
    cout << "\nPlease enter a Day? ";
    cin >> Day;
    return Day;
}

short ReadMonth()
{
    short Month;
    cout << "Please enter a Month? ";
    cin >> Month;
    return Month;
}

short ReadYear()
{
    short Year;
    cout << "Please enter a Year? ";
    cin >> Year;
    return Year;
}

stDate ReadFullDate()
{
    stDate Date;

    Date.Day = ReadDay();
    Date.Month = ReadMonth();
    Date.Year = ReadYear();

    return Date;
}

stDate GetSystemDate()
{
    stDate Date;
    time_t t = time(0);
    tm* now = localtime(&t);

    Date.Year = now->tm_year + 1900;
    Date.Month = now->tm_mon + 1;
    Date.Day = now->tm_mday;

    return Date;
}
```

```cpp
int main()
{
    stDate Date1 = GetSystemDate();

    cout << "\nToday is " << DayShortName(DayOfWeekOrder(Date1))
<< " , "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year <<
endl;

    //---------------------
    cout << "\nIs it End of Week?\n";
    if (IsEndOfWeek(Date1))
        cout << "Yes it is Saturday, it's of Week.";
    else
        cout << "No it's Not end of week.";
    //---------------------
    cout << "\n\nIs it Weekend?\n";
    if (IsWeekEnd(Date1))
        cout << "Yes it is a week end.";
    else
        cout << "No today is " <<
DayShortName(DayOfWeekOrder(Date1)) << ", Not a weekend.";
    //---------------------
    cout << "\n\nIs it Business Day?\n";
    if (IsBusinessDay(Date1))
        cout << "Yes it is a business day.";
   else
        cout << "No it is NOT a business day.";
    //---------------------
    cout << "\n\nDays until end of week : "
        << DaysUntilTheEndOfWeek(Date1) << " Day(s).";
    //---------------------
    cout << "\nDays until end of month : "
        << DaysUntilTheEndOfMonth(Date1) << " Day(s).";

    //---------------------
    cout << "\nDays until end of year : "
        << DaysUntilTheEndOfYear(Date1) << " Day(s).";

    system("pause>0");
    return 0;
}
```