

```

// dtime.cpp - DOUBLE TIME FUNCTIONS
//
// MAINTENANCE HISTORY
// DATE          PROGRAMMER AND DETAILS
// 01-09-08 JS   Original
//
//-----

#include <cstring>           // C-style string manipulation functions
#include <cmath>             // Mathematical functions
#include <cctype>            // Character typing functions
#include <string>            // C++ string declarations
#include <iomanip>           // C++ I/O manipulators
#include <iostream>         // C++ I/O stream declarations
#include <sstream>          // C++ string stream declarations
using namespace std;       // Expand the standard namespace
#include "dtime.h"          // Double time declarations

//-----

// TEST FOR A LEAP YEAR

inline bool
LeapYear (
    int    yr)           // Year number
{
    return yr % 4 != 0 ? 0 : yr % 100 != 0 ? 1 : yr % 400 != 0 ? 0 : 1;
}

//-----

// GET DAY NUMBER FROM CALENDAR YEAR

inline long
DayNoFromYear (
    int    yr)           // Year number
{
    return (static_cast<long>(yr)-1L)*365L+(yr-1)/4-(yr-1)/100+(yr-1)/400;
}

//-----

// PACK A DOUBLE TIME VALUE FROM ITS COMPONENTS

Dtime_t
PackDtime (
    int    yr,           // Year
    int    mo,           // Month
    int    dy,           // Day
    int    hr,           // Hour
    int    mi,           // Minute
    int    se)           // Second
{
    long    dayNo;       // Day number
    const int *dayTbl;   // Pointer to ORD_TBL or LEAP_TBL
    Dtime_t dtime;      // Packed double time value

    // Ordinary and leap year offsets

```

```

static const int ORD_TBL[13] =
    {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365};
static const int LEAP_TBL[13] =
    {0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366};

// Time constants

const static double YOK = - (static_cast<double>(DayNoFromYear(1970L))
    * 24.0 * 60.0 * 60.0);

// Validate the date components

if (yr < 0 || yr >= 3000) return NULL_DTIME;
if (mo < 1 || mo > 12) return NULL_DTIME;

dayTbl = LeapYear(yr) ? LEAP_TBL : ORD_TBL;
dayNo = dayTbl[mo-1] + dy - 1;
if (dy < 0 || dayNo >= dayTbl[mo]) return NULL_DTIME;
dtime = (static_cast<double>(DayNoFromYear(yr))
    + static_cast<double>(dayNo));
dtime = dtime * 24.0 + hr;
dtime = dtime * 60.0 + mi;
dtime = dtime * 60.0 + se;
dtime += YOK;
return dtime;
}

//-----

// UNPACK A DOUBLE TIME VALUE INTO ITS COMPONENTS

void
UnpackDtime (
    Dtime_t      dtime,      // Double time value
    int          *yr,        // Year
    int          *mo,        // Month
    int          *dy,        // Day
    int          *hr,        // Hour
    int          *mi,        // Minutes
    int          *se)        // Seconds
{
    double       x;          // Temporary
    long         dayNo;      // Day number
    long         year;       // Offset year
    int          mth;        // Internal month
    long         n400;       // Number of 400 year periods
    long         n100;       // Number of 100 year periods
    long         n4;         // Number of 4 year periods
    long         n1;         // Number of 1 year periods

// Ordinary and leap year offsets

static const int ORD_TBL[13] =
    {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365};
static const int LEAP_TBL[13] =
    {0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366};

const int      *dayTbl;     // Pointer to ORD_TBL or LEAP_TBL

```

```
// Time constants
```

```
static const double Y0K = - (static_cast<double>(DayNoFromYear(1970))  
    * 24.0 * 60.0 * 60.0);  
static const double Y3K = Y0K +  
    (static_cast<double>(DayNoFromYear(3000)) * 24.0 * 60.0 * 60.0);
```

```
// If the date is outside the acceptable range, return zeros
```

```
if (dtime < Y0K || dtime >= Y3K) {  
    *yr = *mo = *dy = *hr = *mi = *se = 0;  
    return;  
}
```

```
// Transform the time to seconds since 01-01-0001
```

```
dtime -= Y0K;
```

```
// Translate time
```

```
modf ((dtime + 0.5)/60.0, &x);  
*se = (int)( dtime - x*60.0 );  
dtime = x;  
modf ((dtime + 0.5)/60.0, &x);  
*mi = (int)( dtime - x*60.0 );  
dtime = x;  
modf ((dtime + 0.5)/24.0, &x);  
*hr = (int)( dtime - x*24.0 );  
dtime = x;
```

```
// Decode year
```

```
dayNo = (long)(dtime + 0.5);  
n400 = dayNo / DayNoFromYear(400+1);  
dayNo -= n400 * DayNoFromYear(400+1);  
n100 = dayNo / DayNoFromYear(100+1);  
if (n100 >= 4) n100--;  
dayNo -= n100 * DayNoFromYear(100+1);  
n4 = dayNo / DayNoFromYear(4+1);  
dayNo -= n4 * DayNoFromYear(4+1);  
n1 = dayNo / DayNoFromYear(1+1);  
if (n1 >= 4) n1--;  
dayNo -= n1 * DayNoFromYear(1+1);  
year = 400*n400 + 100*n100 + 4*n4 + n1 + 1;  
*yr = year;
```

```
// Calculate day of the year
```

```
dayTbl = LeapYear(year) ? LEAP_TBL : ORD_TBL;  
for (mth = 1; dayTbl[mth] <= dayNo; mth++);  
*mo = mth;  
*dy = dayNo - dayTbl[mth-1] + 1;
```

```
//-----
```

```
// LOAD A DOUBLE TIME VALUE FROM AN SQL DATE OR TIMESTAMP STRING
```

```
Dtime_t
```

```

DtimeFromSql (
    const char *sqlTime) // SQL date or timestamp string
{
    const char *p; // General purpose pointer
    size_t i; // General purpose index
    int yr, mo, dy; // Date components
    int hr, mi, se; // Time components

    p = sqlTime;
    yr = 0;
    for (i = 0; i < 4; i++) {
        if ( ! isdigit(*p)) return NULL_DTIME;
        yr = yr * 10 + *p++ - '0';
    }
    if (*p++ != '-') return NULL_DTIME;
    mo = 0;
    for (i = 0; i < 2; i++) {
        if ( ! isdigit(*p)) return NULL_DTIME;
        mo = mo * 10 + *p++ - '0';
    }
    if (*p++ != '-') return NULL_DTIME;
    dy = 0;
    for (i = 0; i < 2; i++) {
        if ( ! isdigit(*p)) return NULL_DTIME;
        dy = dy * 10 + *p++ - '0';
    }
    if (*p == '-' || *p == ' ') {
        p ++ ;
        hr = 0;
        for (i = 0; i < 2; i++) {
            if ( ! isdigit(*p)) return NULL_DTIME;
            hr = hr * 10 + *p++ - '0';
        }
        if (*p++ != ':' && *p != ':') return NULL_DTIME;
        mi = 0;
        for (i = 0; i < 2; i++) {
            if ( ! isdigit(*p)) return NULL_DTIME;
            mi = mi * 10 + *p++ - '0';
        }
        if (*p++ != ':' && *p != ':') return NULL_DTIME;
        se = 0;
        for (i = 0; i < 2; i++) {
            if ( ! isdigit(*p)) return NULL_DTIME;
            se = se * 10 + *p++ - '0';
        }
        if (*p++ == '.') {
            p ++ ;
            while (isdigit(*p)) p ++ ;
        }
    } else {
        hr = 0;
        mi = 0;
        se = 0;
    }
    if (*p != '\\0') return NULL_DTIME;
    return PackDtime (yr, mo, dy, hr, mi, se);
}

```

//-----

```
// LOAD AN SQL DATE STRING FROM A DOUBLE TIME VALUE
```

```
char *  
SqlDateFromDtime (  
    char      *sqlDate,    // SQL date string  
    Dtime_t   dtime)      // Double time value  
{  
    int      yr, mo, dy; // Date components  
    int      hr, mi, se; // Time components  
    ostringstream  oss;        // Output string stream  
    string      sqlString; // SQL date string  
  
    UnpackDtime (dtime, &yr, &mo, &dy, &hr, &mi, &se);  
  
    oss << setfill('0');  
    oss << setw(4) << yr;  
    oss << '-';  
    oss << setw(2) << mo;  
    oss << '-';  
    oss << setw(2) << dy;  
    sqlString = oss.str();  
    strcpy (sqlDate, sqlString.c_str());  
    return sqlDate;  
}
```

```
//-----
```

```
// LOAD AN SQL TIMESTAMP STRING FROM A DOUBLE TIME VALUE
```

```
char *  
SqlTimestampFromDtime (  
    char      *sqlTimestamp, // SQL date string  
    Dtime_t   dtime)        // Double time value  
{  
    int      yr, mo, dy; // Date components  
    int      hr, mi, se; // Time components  
    ostringstream  oss;        // Output string stream  
    string      sqlString; // SQL timestamp string  
  
    UnpackDtime (dtime, &yr, &mo, &dy, &hr, &mi, &se);  
  
    oss << setfill('0');  
    oss << setw(4) << yr;  
    oss << '-';  
    oss << setw(2) << mo;  
    oss << '-';  
    oss << setw(2) << dy;  
    oss << '-';  
    oss << setw(2) << hr;  
    oss << '.';  
    oss << setw(2) << mi;  
    oss << '.';  
    oss << setw(2) << se;  
    sqlString = oss.str();  
    strcpy (sqlTimestamp, sqlString.c_str());  
    return sqlTimestamp;  
}
```

```
//-----
```

```
// LOAD A DOUBLE TIME VALUE FROM A USER DATE STRING
```

```
Dtime_t  
DtimeFromUserData (  
    const char *userDate) // User date string  
{  
    const char *p; // General purpose pointer  
    int yr, mo, dy; // Date components  
  
    p = userDate;  
    if ( ! isdigit(*p)) return NULL_DTIME;  
    dy = 0;  
    while (isdigit(*p))  
        dy = dy * 10 + *p++ - '0';  
    if (*p++ != '-') return NULL_DTIME;  
    mo = 0;  
    while (isdigit(*p))  
        mo = mo * 10 + *p++ - '0';  
    if (*p++ != '-') return NULL_DTIME;  
    yr = 0;  
    while (isdigit(*p))  
        yr = yr * 10 + *p++ - '0';  
    if (*p != '\\0') return NULL_DTIME;  
    if (yr < 100) {  
        if (yr < 70)  
            yr += 2000;  
        else  
            yr += 1900;  
    }  
    while (*p == ' ') p ++ ;  
    if (*p != '\\0') return NULL_DTIME;  
    return PackDtime (yr, mo, dy, 0, 0, 0);  
}
```

```
//-----
```

```
// LOAD A USER DATE STRING FROM A DOUBLE TIME VALUE
```

```
char *  
UserDataFromDtime (  
    char *userDate, // User date string  
    Dtime_t dtime) // Double time value  
{  
    int yr, mo, dy; // Date components  
    int hr, mi, se; // Time components  
    ostringstream oss; // Output string stream  
    string userString; // User date string  
  
    UnpackDtime (dtime, &yr, &mo, &dy, &hr, &mi, &se);  
  
    oss << setfill('0');  
    oss << setw(2) << dy;  
    oss << '-';  
    oss << setw(2) << mo;  
    oss << '-';  
    oss << setw(2) << (yr % 100);  
    userString = oss.str();  
}
```

```
strcpy (userDate, userString.c_str());  
return userDate;  
}
```

```
//-----
```

```
// LOAD A USER TIME STRING FROM A DOUBLE TIME VALUE
```

```
char *  
UserTimeFromDtime (  
    char *userTime, // User time string  
    Dtime_t dtime) // Double time value  
{  
    int yr, mo, dy; // Date components  
    int hr, mi, se; // Time components  
    ostringstream oss; // Output string stream  
    string userString; // User time string  
  
    UnpackDtime (dtime, &yr, &mo, &dy, &hr, &mi, &se);  
  
    oss << setfill('0');  
    oss << setw(2) << hr;  
    oss << ':';  
    oss << setw(2) << mi;  
    oss << ':';  
    oss << setw(2) << se;  
    userString = oss.str();  
    strcpy (userTime, userString.c_str());  
    return userTime;  
}
```