[算法系列之二十：计算中国农历（一）](http://blog.csdn.net/orbit/article/details/9210413)

世界各国的日历都是以天为最小单位，但是关于年和月的算法却各不相同，大致可以分为三类：

阳历--以天文年作为日历的主要周期，例如：中国公历（格里历）

阴历--以天文月作为日历的主要周期，例如：伊斯兰历

阴阳历--以天文年和天文月作为日历的主要周期，例如：中国农历

我国古人很早就开始关注天象，定昼夜交替为“日”，月轮盈亏为“月”，寒暑交替为“年”，在总结日月变化规律的基础上制定了兼有阴历月和阳历年性质的历法，称为中国农历。本文将介绍中国农历的历法规则、天干地支（Heavenly Stems，Earthly Branches）的计算方法以、二十四节气与中国农历的关系以及知道节气和日月合朔的精确时间的情况下推算中国农历年历的方法。

        在介绍中国农历的历法之前，必须要先介绍一下中国古代的纪年方法。中国古代用天干地支纪年，严格来讲，天干地支纪年以及十二属相并不是中国农历历法的一部分，但是在中国历史上直到今天，天干地支以及十二属相一直都是做为中国农历纪年关系密切的一部分而存在，因此这里先介绍一下天干地支纪年法以及十二属相。

        中国古代纪年不用数字，而是采用天干地支组合。天干有十个，分别是：甲、乙、丙、丁、戊、己、庚、辛、壬、癸；地支有十二个，分别是：子、丑、寅、卯、辰、巳、午、未、申、酉、戌、亥。使用时天干地支各取一字，天干在前，地支在后，组合成干支，例如甲子、乙丑、丙寅等等，依次轮回可形成六十种组合，以这些天干地支组合纪年，每六十年一个轮回，称为一个甲子。实际上中国古代纪月、纪日以及纪时辰都采用干支方法，这些干支组合起来就是我们熟悉的生辰八字。十二属相又称“十二生肖”，由十一种源自自然界的动物：鼠、牛、虎、兔、蛇、马、羊、猴、鸡、狗、猪以及传说中的龙组成，用于纪年时，按顺序和十二地支组合成子鼠、丑牛、寅虎、卯兔、辰龙、巳蛇、午马、未羊、申猴、酉鸡、戌狗和亥猪。天干地支以及十二生肖常组合起来描述农历年，比如公历2011年就是农历辛卯兔年、2012年是壬辰龙年等等。

        计算某一年的天干地支，有很多经验公式，如果知道某一年的天干地支，也可以直接推算其它年份的天干地支。举个例子，如果知道2000年是庚辰龙年，则2012年的干支可以这样推算：（2012-2000）% 10=2，2012年的天干就是从庚开始向后推2个天干，即壬；2012年的地支可以这样推算：（2012 - 2000）% 12 = 0，2012年的地支仍然是辰，因此2012年的天干地支就是壬辰，十二生肖龙年。对于2000年以前的年份，计算出年份差后只要将天干和地支向前推算即可。例如1995年的干支可以这样计算：（2000 – 1995）%10 = 5，（2000 – 1995）%12 = 5，庚向前推算5即是乙，辰向前推算5即是亥，因此1995年的干支就是乙亥，十二生肖猪年。这个干支推算算法的实现如下：

|  |
| --- |
| 202 void CalculateYearGanZhi(int year, int \*gan, int \*zhi)    203 {    204     int sc = year - 2000;    205     \*gan = (7 + sc) % 10;    206     \*zhi = (5 + sc) % 12;    207    208     if(\*gan < 0)    209         \*gan += 10;    210     if(\*zhi < 0)    211         \*zhi += 12;    212 } |

获得2008年的干支纪年：

|  |
| --- |
| 9 TCHAR \*nameOfTianGan[COUNTS\_FOR\_TIANGAN] = {\_T("甲"),\_T("乙"),\_T("丙"),\_T("丁"),\_T("戊"),\_T("己"),\_T("庚"),\_T("辛"),\_T("壬"),\_T("癸") };     10 TCHAR \*nameOfDiZhi[COUNTS\_FOR\_DIZHI] = {\_T("子"),\_T("丑"),\_T("寅"),\_T("卯"),\_T("辰"),\_T("巳"),\_T("午"),\_T("未"),\_T("申"),\_T("酉"),\_T("戌"),\_T("亥")};    146     int gan,zhi;    147    148     CalculateYearGanZhi(2008, &gan, &zhi);    149    150     text.Format(\_T("农历【%s%s】%s年"),    151                 year, m\_curMonth, nameOfTianGan[gan - 1], nameOfDiZhi[zhi - 1], nameOfShuXiang[zhi- 1]); |

结果是：农历戊子鼠年。

        中国农历是以月亮运行周期为基础，结合太阳运行规律（二十四节气）制定的历法，农历月的定义规则就是中国农历历法的关键，因此要了解中国农历的历法规则，就必须知道如何定义月，如何设置闰月？中国农历的一年有十二个月或十三个月，但是正统的叫法只有十二个月，分别是正月、二月、三月、四月、五月、六月、七月、八月、九月、十月、冬月和腊月（注意，正统的中国农历是没有十一月和十二月的，如果你用的历法软件有显示农历十一月和农历十二月，就说明非常不专业）。中国民间常用“十冬腊月天”来形容寒冷的天气，其实指的就是十月，十一月和十二月这三个最冷的月份。一年有十三个月的情况是因为有闰月，多出来的这个闰月没有月名，只是跟在某个月后面，称为闰某月。比如公历2009年对应的农历乙丑年，就是闰五月，于是这一年可以过两个端午节。

        中国农历为什么会有闰月？其实中国农历置闰月是为了协调回归年和农历年的矛盾。前面提到过，中国农历是一种阴阳历，农历的月分大月和小月，大月一个月是30天，小月一个月是29天。中国农历把日月合朔（太阳和月亮的黄经相同，但是月亮不可见）的日期定位月首，也就是“初一”，把月圆的时候定为望日，也就是“十五”，月亮绕地球公转一周称为一个朔望月。天文学的朔望月长度是29.5306日，中国农历以朔望月为基础，严格保证每个月的头一天是朔日，这就使得每个月是大月还是小月的安排不能固定，通常需要通过天文学观测和计算来确定。一个农历年由12个朔望月组成，这样一个农历年的长度就是29.5306  12 = 354.3672日，而阳历的一个天文学回归年是365.2422日，这样一个农历年就比一个回归年少10.88天，这个误差如果累计起来过16年就会出现“六月飞雪”的奇观了。为了协调农历年和回归年之间的矛盾，聪明的先人在天文观测的基础上，找到了“闰月”的方法，通过在适当的月份插入闰月来保证每个农历年的正月到三月是春季，四月到六月是夏季，七月到九月是秋季，十月到十二月是冬季，也就是说，让历法和天文气象能够基本对上，不至于出现“六月飞雪”。

        那么多长时间增加一个闰月比较合适呢？最早人们推算是“三年一闰”，后来是“五年两润”，随着历法计算的精确，最终定型为“十九年七闰”。这个“十九年七闰”又是怎么算出来的呢？其实就是求出回归年日数和朔望月日数的最小公倍数，也就是m个回归年的天数和n个朔望月的天数相等，即：

m  365.2422 = n  29.5306

这样m和n的比例就是29.5306 : 365.2422  19 : 235，按照这个最接近的整数倍数关系，每19个回归年需要添加的闰月就是：

235 – 12 \* 19 = 7

也就是“十九年七闰”的由来。但是需要注意的是，“十九年七闰”也并不是精确的结果，每19年就会有0.0892天的误差：

19 \* 365.2422 - 235 \* 29.5306 =  0.0892

这样每213年就会积累约1天的误差，因此，即使按照“十九年七闰”计算，中国农历每一两百年就需要修正一次。正因为这样，现行农历从唐代以后就已经不再遵守“十九年七闰”法，而是采用更准确的“中气置闰”法。“中气置闰”法更准确的名称应该是“定冬至”法，就是定两个冬至节气之间的时间为一个农历年，这样农历年的长度就和太阳回归年长度对应，不会产生误差。

        现在，我们知道农历通过置闰月的方式协调农历年和回归年长度不相等的问题，也知道了置闰的方法是“中气置闰”法，那么到底什么是“中气”，又是如何定中气置闰月呢？要回答这个问题，就需要介绍另一个天文现象――节气。二十四节气起源于黄河流域，远在春秋时代，就定出仲春、仲夏、仲秋和仲冬等四个节气。以后不断地改进与完善，到秦汉年间，二十四节气已完全确立，汉武帝太初元年（公元前104年）制定的《太初历》，则第一次从历法上明确了二十四节气的天文位置。

        地球沿着一个近似椭圆轨道绕太阳公转，这个公转轨道所在的平面就是“黄道面”，黄道面向外延伸与天球的交线就是“黄道”。古人由于观测条件限制，只能根据视觉感觉认为是太阳沿着黄道绕地球运转，因此设定太阳从黄经（黄道经度）零度起（以春分点为起点自西向东度量），将太阳沿黄经每运行15度所经历的时日称为“一个节气”。太阳每年运行360度，共经历二十四个节气，春季的节气有立春（315度）、雨水（330度）、惊蛰（345度）、春分（0度、360度）、清明（15度）和谷雨（30度），夏季的节气有立夏（45度）、小满（60度）、芒种（75度）、夏至（90度）、小暑（105度）和大暑（120度），秋季的节气有立秋（135度）、处暑（150度）、白露（165度）、秋分（180度）、寒露（195度）和霜降（210度）。冬季的节气有立冬（225度）、小雪（240度）、大雪（255度）、冬至（270度）、小寒（285度）和大寒（300度）。二十四节气又细分为十二节气和十二中气，二十四节气按照顺序排在奇数位置上的就是节气，排在偶数位置上的就是中气。也就是说，立春、惊蛰、清明、立夏、芒种、小暑、立秋、白露、寒露、立冬、大雪和小寒就是十二个节气，而雨水、春分、谷雨、小满、夏至、大暑、处暑、秋分、霜降、小雪、冬至和大寒就是十二个中气。二十四个节气平分在公历的12个月中，每月一节气一中气。二十四节气反映了太阳的周年运动（以地球为参照物的视运动），所以节气在现行的公历中日期基本固定，上半年在6日、21日，下半年在8日、23日，前后不差 1～2天。中国民间流传的《二十四节气歌》就是为了方便记忆这些节气：

春雨惊春清谷天，

夏满芒夏暑相连，

秋处露秋寒霜降，

冬雪雪冬小大寒，

每月两节不变更，

最多相差一两天。

传统上一个农历年起于冬至，终于冬至，因此要确定在哪一年置闰，主要看那一年两个冬至之间有几个朔望月，如果是12个朔望月，则不置闰，如果是十三个朔望月，则置闰月，至于闰几月，则要看节气而定。对于有13个朔望月的农历年，置闰月的规则就是从农历二月开始到十月，第一个没有中气的月就是闰月，这个没有中气的朔望月跟在哪个月后面就是闰几月。为什么会有没有中气的朔望月呢？黄道上两个中气之间相隔30度，一个回归年的长度是365.2422日，则两个中气之间的平均间隔是365.2422 12 = 30.4368日，但是因为地球轨道是椭圆轨道，因此相邻的两个中气的时间间隔是不均匀的，比如在远地点附近的中气间隔就会长一点，最长可能是31.45天。而农历的朔望月平均长度是29.5306日，这样就会出现某个朔望月刚好落在两个中气之间的情况，比如，某个月的上一个月月末是一个中气，但是下一个中气落在这个月的下一个月的头几天里，这样这个月就没有中气了。举个例子，2001年农历辛已年的四月二十九（公历5月21日）是小满，农历四月之后的这个朔望月从公历5月23日持续到公历6月20日，而小满后的下一个中气夏至是在公历的6月21日，也就是农历四月的下下个月的初一，这样农历四月后的这个月就没有中气，跟在四月之后，就称为闰四月。

        由于节气在回归年中是均匀分布的，因此公历中的节气日期基本上是固定的，比如立春是在公历的2月3－5日，不会超出这个日期范围，这也就是《二十四节气歌》所说的：每月两节不变更，最多相差一两天。但是在中国农历中哪个中气属于哪个月是有规定的，雨水是正月的中气，春分是二月的中气，谷雨是三月的中气，小满是四月的中气，夏至是五月的中气，大暑是六月的中气，处暑是七月的中气，秋分是八月的中气，霜降是九月的中气，小月是十月的中气，冬至是十一月的中气，大寒是十二月的中气。

        在了解了农历与节气的关系以及农历如何置闰月的方法之后，还需要解决一个问题才能着手农历年历的推算，那就是如何确定农历年的开始，或者说哪个月的初一是农历新年的开始？要回答这个问题，就需要了解中国农历特有的“月建”问题。

        中国农历是阴阳合历，需要同时考虑太阳和月亮的位置。所以在确定岁首（元旦）时，需要先确定它在某个季节，然后再选定与这个季节相近的朔望月作为岁首。由于一岁（一个回归年）和12个阴历月并不相等，相差约10.88天，因此每隔三年需要设置一个闰月调整季节。中国上古的天文学家想出了一个简便的方法判断月序与季节的关系，这就是以傍晚时北斗七星的斗柄的指向确定月序，称为“十二月建”。从北方起向东转，将地面划分为十二个方位，傍晚时北斗所指的方位，就是该月的月建，其子月为冬至所在之月，对应十一月，丑月是冬至所在之月的次月，对应十二月，寅月在丑月之后，对应正月。中国在历史上的不同时期，多次修改过岁首（元旦）的起始月份，上古时代就有“三正”之说，所谓“三正”，就是“夏正建寅、殷正建丑、周正建子”，意思是夏历以寅月（正月）为岁首，殷历以丑月（十二月）为岁首，周历以子月（十一月）为岁首。从秦代到西汉前期又采用秦历，秦历建亥，也就是以亥月作为岁首之月，汉武帝太初元年（公元104年）改用太初历，重新适用建寅的夏历，以寅月（正月）为岁首。在这之后的两千多年时间里，除王莽和魏明帝一度改用建丑的殷历，唐武后和肃宗时改用建子的周历外，各个朝代均使用建寅的夏历直到清朝末年。辛亥革命胜利以后，南京国民政府将公历1月1日改为元旦，但是人们仍习惯称农历的正月初一为元旦。新中国成立初期召开的第一届政治协商会议，正式将公历的1月1日确定为元旦，将农历的正月初一定为“春节”，也就是说，农历的岁首仍然采用夏历从寅月（正月）开始。

        了解了“月建”问题，就解决了农历朔望月与公历月的对应关系，那就是冬至节气所在的朔望月就是农历的子月，对于目前适用的夏历建寅的月建体系，就意味着冬至节气所在的朔望月是农历的十一月，只要找到这个朔望月的起始日（日月合朔发生的时刻所在的那一日），就找到了公历的日期与农历日期的对应关系。下面总结一下中国农历历法的基本法则：

1、严格以日月合朔发生时刻为月首，这一天定为初一，通过计算两次日月合朔的时间间隔确定每月是29天还是30天；

2、月以中气得名，冬至节气总是出现在农历十一月，包含雨水中气的月为正月（即寅月），月无中气者为闰月，与前一个月同名；

3、从某一年的冬至后第一天开始，到下一个冬至这段时间内，如果有十三个朔望月出现，则此期间要增加一个闰月，从二月到十月，第一个没有中气的月就是闰月，如果在此期间有超过两个朔望月没有中气，则只有第一个没有中气的朔望月是闰月；

4、农历年以正月初一为岁首（关于农历岁首的说法，请参考文末附加的《小知识5：正月初一和立春节气》），以腊月（十二月）廿九或三十为除夕；

5、如果节气和日月合朔在同一天，则该节气是这个新朔望月的节气。（民间历法）

        规则5对节气和朔日在同一天的处理，采用了民间历法的处理原则，关于民间历法和历理历法的区别，请参考文末附加的《小知识1：民间历法和历理历法》。

        了解了农历历法的基本法则后，就可以根据历法进行农历年历的推算。农历年历的推算是一件很复杂的事情，需要知道每年二十四个节气和本年内每次日月合朔的精确时间，这些时间的获取比较困难。现在有很多可以显示农历的日历软件，其实并不计算这些时间，而是事先从权威机构（如紫金山天文台）获取这些经过推算的时间，然后用各种方法将这些信息存储在设计好的数据结构中。当计算农历时采用查表的方法获取每年的二十四节气日期、大小月情况以及闰月情况，这样的软件受数据量的限制，往往只能显示近一两百年的年历。

        还有一种确定节气时间和朔日时间的方法，就是在已知某个节气或朔日的精确时间后，通过某些规律向前或向后推算其它节气或朔日的时间。有一些经验公式可以用来计算节气发生的日期，比如“通式寿星公式”，可以计算出某一年的某个节气时间，但是只能精确到日。关于“通式寿星公式”的详细内容，请参考文末附加的《小知识2：通式寿星公式》。至于精确的节气或朔日时间，也只能从权威机构获取。以节气的时间推算为例，二十四个节气就是黄道上的24各点，由于地球运动受其它天体的影响，导致这些节气在每年的时间是不固定的，但是这些节气之间的间隔时间基本上可以看作是固定的，下表就是二十四节气的时间间隔表：

|  |  |  |
| --- | --- | --- |
| 节气名 | 与上一节气之间的时间差 | 与小寒节气的累积时间差 |
| 小寒 | 1271448.00 | 0.00 |
| 大寒 | 1272494.40 | 1272494.40 |
| 立春 | 1275526.20 | 2548020.60 |
| 雨水 | 1282123.20 | 3830143.80 |
| 惊蛰 | 1290082.80 | 5120226.60 |
| 春分 | 1300639.20 | 6420865.80 |
| 清明 | 1311153.00 | 7732018.80 |
| 谷雨 | 1323253.80 | 9055272.60 |
| 立夏 | 1333685.40 | 10388958.00 |
| 小满 | 1344107.40 | 11733065.40 |
| 芒种 | 1351227.00 | 13084292.40 |
| 夏至 | 1357299.60 | 14441592.00 |
| 小暑 | 1358968.80 | 15800560.80 |
| 大暑 | 1358786.40 | 17159347.20 |
| 立秋 | 1354419.00 | 18513766.20 |
| 处暑 | 1348236.00 | 19862002.20 |
| 白露 | 1339003.20 | 21201005.40 |
| 秋分 | 1328654.40 | 22529659.80 |
| 寒露 | 1317185.40 | 23846845.20 |
| 霜降 | 1305760.80 | 25152606.00 |
| 立冬 | 1295081.40 | 26447687.40 |
| 小雪 | 1285764.00 | 27733451.40 |
| 大雪 | 1278469.80 | 29011921.20 |
| 冬至 | 1273556.40 | 30285477.60 |

表（1）二十四节气时间间隔表（单位：秒钟）

已知1900年小寒时刻为1月6日2:05:00，以这个节气时刻为基准，推算其它年份节气的算法实现如下：

|  |
| --- |
| 8 static double s\_stAccInfo[] =      9 {     10     0.00, 1272494.40, 2548020.60, 3830143.80, 5120226.60, 6420865.80,     11     7732018.80, 9055272.60, 10388958.00, 11733065.40, 13084292.40,14441592.00,     12     15800560.80, 17159347.20, 18513766.20, 19862002.20, 21201005.40,22529659.80,     13     23846845.20, 25152606.00, 26447687.40, 27733451.40, 29011921.20,30285477.60     14 };     15     16 //已知1900年小寒时刻为1月6日02:05:00     17 const double base1900\_SlightColdJD = 2415025.5868055555;     18     19 double CalculateSolarTermsByExp(int year, int st)     20 {     21     if((st < 0) || (st > 24))     22         return 0.0;     23     24     double stJd = 365.24219878 \* (year - 1900) + s\_stAccInfo[st] / 86400.0;     25     26     return base1900\_SlightColdJD + stJd;     27     28 } |

base1900\_SlightColdJD是北京时间1900年1月6日凌晨2:05:00的儒略日数，CalculateSolarTermsByExp()函数返回指定年份的节气的儒略日数。已知某个朔日的精确时间推算其它朔日时间的方法也类似，以朔望月的长度为单位向前或向后累加即可。

        这种推算的方法是建立在地球回归年的长度是固定365.2422天、节气的间隔是绝对固定的、朔望月长度是平均的29.5305天等假设之上的，由于天体运动的互相影响，这种假设不是绝对成立的，因此这种推算方法的误差很大。以CalculateSolarTermsByExp()函数为例，计算1900年前后30年内的节气时间的误差还可以控制在30分钟以内，但是到2000年的时候误差已经超过130分钟了。人们还总结出了计算节气和朔日时间的两个经验公式，本文末尾附加的《小知识3：计算节气和朔日的经验公式》一节会详细介绍这两个公式，不过这两个公式的结果也只能精确到日，不能提供10秒以内精度的时间。要想精确地获得几千年乃至更长时间范围内任意一年的节气发生时间和日月合朔时间，就只能采用“天文算法”。

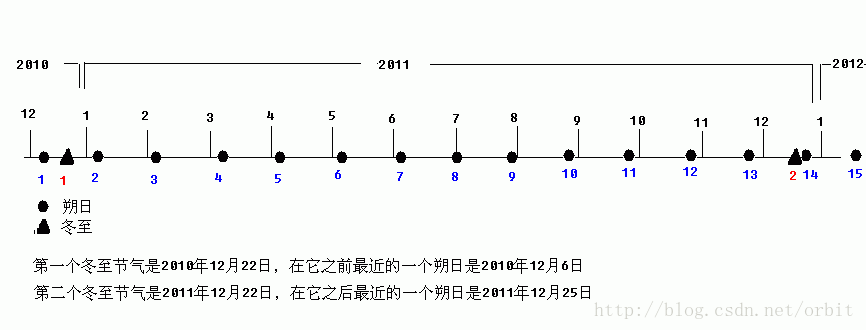
《继续：天文算法计算农历。。。》

[算法系列之二十：计算中国农历（二）](http://blog.csdn.net/orbit/article/details/9337377)

（接上篇）

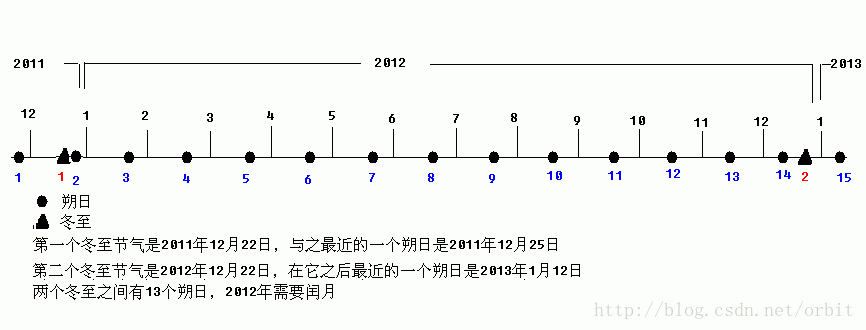
        所谓的“天文算法”，就是利用经典力学定律推导行星运转轨道，对任意时刻的行星位置进行精确计算，从而获得某种天文现象发生时的时间，比如日月合朔这一天文现象就是太阳和月亮的地心黄经（视黄经）差为0的那一瞬间。能够计算任意时刻行星位置的一套理论就被称为星历表，比较著名的星历表有美国国家航空航天局下属的喷气推进实验室发布的DE系列星历表，还有瑞士天文台在DE406基础上拓展的瑞士星历表等等。根据行星运行轨道直接计算行星位置通常不是很方便，更何况大多数民用天文计算用不上那么多精确的轨道参数，于是天文学家在这些星历表的基础上推导出了很多可以做简便计算，但是又能保证一定精度的行星运行理论，比较著名的有VSOP82/87太阳系行星运行理论和ELP-2000/82月球运行理论，这两套理论在精度上已经很接近DE系列星历表了。关于如何应用这两套伦理进行天文历法计算，请参考“日历生成算法”系列文章的第三篇《用天文方法计算二十四节气》和第四篇《用天文方法计算日月合朔》，本文介绍的农历年历推算是在已经通过天文算法获得了精确的节气时间和日月合朔时间的基础上进行的。

        中国的官方纪时采用的是中国公历（格里历），因此农历年历的推导应以公历年的周期为主导，附上农历年的信息，也就是说，年历以公历的1月1日为起始，至12月31日结束，根据农历历法推导出的农历日期信息，附加在公历日期信息上形成双历。通常情况下，一个公历年周期都不能完整地对应到一个农历年周期上，二者的偏差也不固定，因此不存在稳定的对应关系，也就是说，不存在从公历的日期到农历日期的转换公式，只能根据农历的历法规则推导出农历日期与公历日期的对应关系。由农历历法规则可知，上一个公历年的冬至（）所在的朔望月是上一个农历年的十一月（冬月），所以在进行节气计算时，需要计算包括上一年冬至节气在内的二十五个节气，才能对应上上一个农历年的十一月和当前农历年的十一月。在计算与之对应的朔日时，考虑到有闰月的情况，需要从上一年冬至节气前的第一个朔日，连续计算15个朔日才能保证覆盖两个冬至之间的一整年时间，图（1）显示了2011年没有闰月的情况下朔日和冬至的关系：



图（1）没有闰月情况下朔日与冬至节气关系图

图中上排数字是公历月的编号，黑色圆点代表朔日，黑色三角形代表冬至节气。图（2）显示了2012年有闰月的情况下朔日和冬至的关系：



图（2）有闰月情况下朔日与冬至节气关系图

通过计算得到能够覆盖两个冬至节气的所有朔日时间后，就可以着手建立公历日期与农历日期的对应关系。以图（1）所示的2011年为例，首先根据计算得到的15个朔日（2011年只会用到其中的前14个时间）时间，建立与2011年（公历年）有关的朔望月关系表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 朔日编号 | 合朔时间 | 对应公历日期 | 月长 | 月名 |
| 1 | 01:35:39.90 | 2010-12-06 | 29 | 冬月 |
| 2 | 17:02:34.26 | 2011-01-04 | 30 | 腊月 |
| 3 | 10:30:42.67 | 2011-02-03 | 30 | 正月 |
| 4 | 04:45:59.44 | 2011-03-05 | 29 | 二月 |
| 5 | 22:32:15.13 | 2011-04-03 | 30 | 三月 |
| 6 | 14:50:31.79 | 2011-05-03 | 30 | 四月 |
| 7 | 05:02:32.51 | 2011-06-02 | 29 | 五月 |
| 8 | 16:53:54.10 | 2011-07-01 | 30 | 六月 |
| 9 | 02:39:45.06 | 2011-07-31 | 29 | 七月 |
| 10 | 11:04:06.43 | 2011-08-29 | 29 | 八月 |
| 11 | 19:08:50.09 | 2011-09-27 | 30 | 九月 |
| 12 | 03:55:54.64 | 2011-10-27 | 29 | 十月 |
| 13 | 14:09:40.97 | 2011-11-25 | 30 | 冬月 |
| 14 | 02:06:27.05 | 2011-12-25 | 29 | 腊月 |
| 15 | 15:39:23.99 | 2012-01-23 | 30 | 正月 |

表（2）2011年朔望月与公历日期关系表

编号为1和2的两个朔日之间的朔望月是十一月，因为冬至节气落在这个朔望月，其它月的月名依次类推，正月的朔日就是春节。输出公历和农历双历时，以月（公历）为单位，从每月第一天开始，依次判断每一天属于哪个朔望月，确定这一天的农历月名，然后比较这一天和这个朔望月的朔日之间相差几天，记为农历日期。以2011年1月1日为例，这一天在2010年12月6日（2010年农历十一月的朔日）和2011年1月4日之间（2010年农历十二月的朔日），查表（1）可知对应的农历月是十一月，这一天和2010年12月6日相差26天，因此这一天的农历日期就是“廿七”。再以2011年2月3日（春节）这一天为例，查朔望月表得知2月3日属于从2月3日开始的朔望月，这个朔望月的月名是正月，而2月3日就是月首，农历日期是初一，正月初一就是春节。

先来介绍两个函数，这两个函数分别用于计算节气和日月合朔发生的时间，函数算法的具体描述将在“日历生成算法”系列文章的第三篇《用天文方法计算二十四节气》和第四篇《用天文方法计算日月合朔》中介绍，此处只是简单介绍一下用法。首先是计算节气时间的函数：

|  |
| --- |
| 5 double CalculateSolarTerms(int year, int angle); |

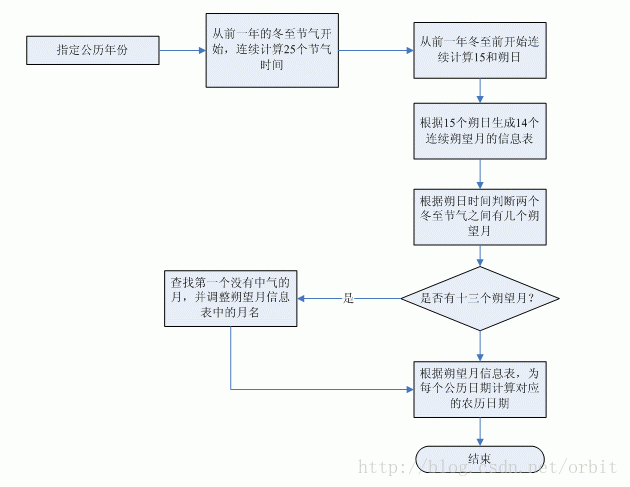
这个函数用于计算指定的年份（year参数）中，太阳在黄道上运行（视运动）到指定角度时的时间，angle可以设定节气发生时的角度，比如CalculateSolarTerms(2011, 270)就是计算2011年冬至的时间。这个函数返回的时间类型是儒略日，关于儒略日的说明请参考“日历生成算法”系列文章的第一篇《中国公历（格里历）》。

        接下来介绍计算日月合朔时间的函数：

|  |
| --- |
| 8 double CalculateMoonShuoJD(double tdJD); |

这个函数返回指定时间附近的朔日时间，搜索的范围是tdJD参数指定时间的前一天到后29.5305天，tdJD参数和返回值的时间类型都是儒略日。

        生成指定公历年份的公历和农历的双历年历的流程如下：



图（3）计算公农历双历年历的流程

GetAllSolarTermsJD()函数从指定年份的指定节气开始，连续计算25个节气时间，时间可以跨年份，内部判断过冬至节气后自动转到下一年的节气继续计算：

|  |
| --- |
| 139 void CChineseCalendar::GetAllSolarTermsJD(int year, int start, double\*SolarTerms)    140 {    141     int i = 0;    142     int st = start;    143     while(i < 25)    144     {    145         double jd = CalculateSolarTerms(year, st \* 15);    147         if(st == WINTER\_SOLSTICE)    148         {    149             year++;    150         }    151         st = (st + 1) % SOLAR\_TERMS\_COUNT;    152     }    153 } |

start参数是节气的索引，定义二十四节气的索引如下：

|  |
| --- |
| 38 const int VERNAL\_EQUINOX      = 0;    // 春分     39 const int CLEAR\_AND\_BRIGHT    = 1;    // 清明     40 const int GRAIN\_RAIN          = 2;    // 谷雨     41 const int SUMMER\_BEGINS       = 3;    // 立夏     42 const int GRAIN\_BUDS          = 4;    // 小满     43 const int GRAIN\_IN\_EAR        = 5;    // 芒种     44 const int SUMMER\_SOLSTICE     = 6;    // 夏至     45 const int SLIGHT\_HEAT         = 7;    // 小暑     46 const int GREAT\_HEAT          = 8;    // 大暑     47 const int AUTUMN\_BEGINS       = 9;    // 立秋     48 const int STOPPING\_THE\_HEAT   = 10;   // 处暑     49 const int WHITE\_DEWS          = 11;   // 白露     50 const int AUTUMN\_EQUINOX      = 12;   // 秋分     51 const int COLD\_DEWS           = 13;   // 寒露     52 const int HOAR\_FROST\_FALLS    = 14;   // 霜降     53 const int WINTER\_BEGINS       = 15;   // 立冬     54 const int LIGHT\_SNOW          = 16;   // 小雪     55 const int HEAVY\_SNOW          = 17;   // 大雪     56 const int WINTER\_SOLSTICE     = 18;   // 冬至     57 const int SLIGHT\_COLD         = 19;   // 小寒     58 const int GREAT\_COLD          = 20;   // 大寒     59 const int SPRING\_BEGINS       = 21;   // 立春     60 const int THE\_RAINS           = 22;   // 雨水     61 const int INSECTS\_AWAKEN      = 23;   // 惊蛰 |

节气索引乘以15就是节气在黄道上对应的度数。GetNewMoonJDs()函数从指定时间开始连续计算15个朔日时间，从第一个冬至节气前的第一个朔日开始。15个朔日可以形成14个完整的朔望月，保证在有闰月的情况下也能包含两个冬至节气：

|  |
| --- |
| 137 void CChineseCalendar::GetNewMoonJDs(double jd, double \*NewMoon)    138 {    139     for(int i = 0; i < NEW\_MOON\_CALC\_COUNT; i++)    140     {    141         double shuoJD = CalculateMoonShuoJD(jd);    142         NewMoon[i] = shuoJD;    143    144         jd += 29.5; /\*转到下一个最接近朔日的时间\*/    145     }    146 } |

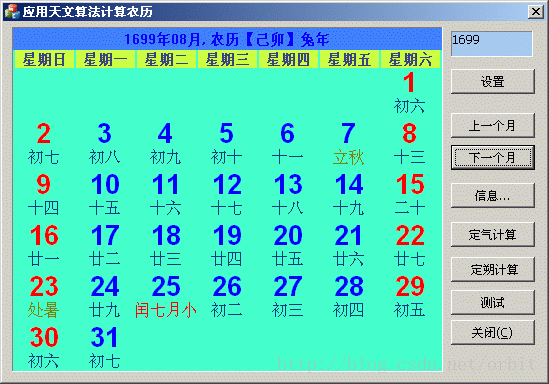
BuildAllChnMonthInfo()函数根据15个朔日时间组成14个朔望月，根据相邻朔日的间隔计算出农历月天数用来判定大小月，并且从“十一月”开始依次为每个朔望月命名（月建名称）：

|  |
| --- |
| 170 bool CChineseCalendar::BuildAllChnMonthInfo()    171 {    172     CHN\_MONTH\_INFO info; //一年最多可13个农历月    173     int i;    174     int yuejian = 11;   //采用夏历建寅，冬至所在月份为农历11月    175     for(i = 0; i < (NEW\_MOON\_CALC\_COUNT - 1); i++)    176     {    177         info.mmonth = i;    178         info.mname = (yuejian <= 12) ? yuejian : yuejian - 12;    179         info.shuoJD = m\_NewMoonJD[i];    180         info.nextJD = m\_NewMoonJD[i + 1];    181         info.mdays = int(info.nextJD + 0.5) - int(info.shuoJD + 0.5);    182         info.leap = 0;    183    184         CChnMonthInfo cm(&info);    185         m\_ChnMonthInfo.push\_back(cm);    186    187         yuejian++;    188     }    189    190     return (m\_ChnMonthInfo.size() == (NEW\_MOON\_CALC\_COUNT - 1));    191 } |

CalcLeapChnMonth()函数根据节气和朔日时间判断在两个冬至节气之间的农历年是否有闰月，判断的依据就是看第十四个朔日是否在第二个冬至节气之前，如果第十四个朔日发生在第二个冬至节气之前，就说明在两个冬至节气之间发生了十三次朔日，需要置闰月。因为农历中十二个中气属于哪个农历月是固定的，因此置闰月的过程就是依次判断十二个中气是否在对应的农历月中，如果本应该属于某个农历月的中气却没有落在这个农历月中，则这个农历月就是闰月，需要设置闰月标志，同时调整这个月之后的月名。调整农历月名的方法就是月名减一，比如原来是八月就要调整为七月，这样就将十三个月对应上了十二个月名（其中多出来的一个农历月被命名为闰某月）。如果节气和朔日发生在同一天，CalcLeapChnMonth()函数采用的是民间历法的规则，与现行历法一致：

|  |
| --- |
| 194 void CChineseCalendar::CalcLeapChnMonth()    195 {    196     assert(m\_ChnMonthInfo.size() > 0); /\*阴历月的初始化必须在这个之前\*/    197    198     int i;    199    200     if(int(m\_NewMoonJD[13] + 0.5) <= int(m\_SolarTermsJD[24] + 0.5)) //第13月的月末没有超过冬至，说明今年需要闰一个月    201     {    202         //找到第一个没有中气的月    203         i = 1;    204         while(i < (NEW\_MOON\_CALC\_COUNT - 1))    205         {    206    207             /\*m\_NewMoonJD[i + 1]是第i农历月的下一个月的月首，本该属于第i月的中气如果比下一个月    208               的月首还晚，或者与下个月的月首是同一天（民间历法），则说明第i月没有中气\*/    209             if(int(m\_NewMoonJD[i + 1] + 0.5) <= int(m\_SolarTermsJD[2 \* i] +0.5))    210                 break;    211             i++;    212         }    213         if(i < (NEW\_MOON\_CALC\_COUNT - 1)) /\*找到闰月，对后面的农历月调整月名\*/    214         {    215             m\_ChnMonthInfo[i].SetLeapMonth(true);    216             while(i < (NEW\_MOON\_CALC\_COUNT - 1))    217             {    218                 m\_ChnMonthInfo[i++].ReIndexMonthName();    219             }    220         }    221     }    222 } |

        从理论上讲，本文介绍的算法在精度允许的范围内可以计算前后几千年的农历年历，但是对古代的农历计算需要小心。首先是“平朔”和“定朔”的问题，唐代以前使用的是平朔方法定月首，本文介绍的计算方法采用的是“定朔”方法，因此计算出的年历与唐代以前的历史会不一致。另外，即是在唐代以后采用“定朔”的历法，因为古代天文观测和计算受条件限制，可能不够精确，因此与现在用天文算法计算出的结果可能并不一致。所以对历史农历的计算应该以历史事实为主，天文计算为辅，当计算与历史不一致时，要根据历史数据进行校正。Calendar.exe是根据本文介绍的算法编写的日历小程序，没有太多的功能，主要是为了验证算法，因为没有历史数据用于修正结果，因此不支持1601年以前的农历计算（也就是说按照天文算法计算出来的结果可能和实际历史上的历法不符）。



图（5）演示程序的界面

**小知识1：民间历法和历理历法**

    新中国成立以后没有颁布新的“官方农历历法”，将历法和政治分离体现了时代的进步，但是由于没有 “官方历法”，也引起了一些问题。比如我国现在采用的农历历法是《时宪历》，它源于清朝顺治年间（公元1645）颁布的《顺治历》，它有两个不足之处：一个是日月合朔和节气的时间以北京当地时间为准，也就是东经116度25分的当地时间，其节气和新月的观察只适用于中原地区。其它经度的地方，因为时间的关系，对导致日月合朔和节气时间的差异导致置闰和月顺序各不相同。另一个不足之处就是日月合朔时间和节气时间判断不精确，如果日月合朔时间和节气时间在同一天，不管具体的时间是否有先后，一律将此节气算做新月中的节气，这样一来，如果这个节气是中气，就会影响到闰月的设置。历理历法针对这两点进行了改进，对节气时间和日月合朔时间统一采用东经120度即东八区标准时，这样在任何时区的节气和置闰结果都是一样的，以东八区标准时为准。对于节气时间和日月合朔时间在同一天的情况，精确计算到时、分、秒，只有日月合朔时间在节气时间之前，这个节气才包含在次月内。历理历法从理论上讲更符合现代天文学的精确计算，但是需要注意的是，历理历法仍然只是存在于理论上的历法，我国现行的农历历法依然是民间历法《时宪历》或《顺治历》。

**小知识2：通式寿星公式**

“通式寿星公式”是前人整理出来的一个用于计算每年立春日期的经验公式:

Date = 向下取整(Y \* D + C) - L

其中，Y是年份，D的值是0.2422，C是经验值，取决于节气和年份，对于21世纪，立春节气的C值是4.475，春分节气的C值是20.646等等；

L是闰年数，其计算公式为：

L = 向下取整(Y/4) - 向下取整(Y/100) + 向下取整(Y/400)

用“通式寿星公式”确定2011年立春日期的过程如下：

L = int(2011/4) – int(2011/100) + int(2011/400) = 502 – 20 + 5 = 487

Date = int（2011×0.2422+4.475）- 487 = 491 – 487 = 4

所以，2011年的立春日期是2月4日。

**小知识3：计算节气和朔日的经验公式**

    以1900年1月0日（星期日）为基准日，之后的每一天与基准日的差值称为“积日”， 1900年1月1日的积日是1，以后的时间依次类推，则计算第y年第x个节气的积日公式是：

F = 365.242 \* (y – 1900) + 6.2 + 15.22 \*x - 1.9 \* sin(0.262 \* x)

其中x是节气的索引，0代表小寒，1代表大寒，其它节气按照顺序类推。

计算从1900年开始第m个朔日的公式是：

M = 1.6 + 29.5306 \* m + 0.4 \* sin(1 - 0.45058 \* m)

**小知识4：平朔和定朔**

    中国农历的朔望月长度是平均29.5305天，所以农历月就有大月30天，小月29天之分，从先秦时期到唐代，农历历法均是采用大小月轮流交替的方式设置每个农历月的天数，只有少数情况下才出现连续两个大月的情况，采用这种方式的历法就称为“平朔”。“平朔”历法简单，但是不能保证日月合朔发生在初一这一天，有可能是上月的月末一天，也有可能是本月初二。南北朝时期，一种新的历法被提出来，这种历法严格按照日月合朔为月初制定农历月，采用这种方式的历法就称为“定朔”。“定朔”历法严格将日月合朔时间确定月初，因为月球公转是椭圆轨道，速度并不是均匀，所以会发生连续多个大月或连续多个小月的情况，导致“定朔”历法推广遇到很大的阻力，直到唐代，中国历法才全面弃用“平朔”，改用“定朔”。

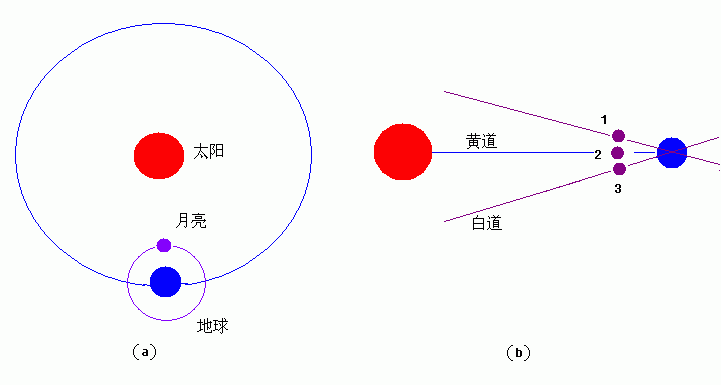
**小知识5：正月初一和立春节气**

    立春是二十四节气之首，所以古代民间都是在“立春”这一天过节，相当于现代的春节（中国古代即是节气也是节日的情况很多，比如清明、冬至等等）。1911年，孙中山领导的辛亥革命建立了中华民国，在从历法上正式把农历正月初一定为“春节”，把公历1月1日定为“元旦”，也就是“新年”。农历年从正月初一开始没有争议，但是农历生肖年从何时开始却一直有争议，目前多数人都认为“立春”节气是农历生肖年的开始。因为在中国古代历法中，十二生肖的计算与天干地支有很大关系，所以在“论天干地支、计算廿四节气”的情况下，“立春”节气应该是新生肖的开始。对于普通老百姓来说，习惯于认为正月初一是生肖年的开始，因此，正月初一和“立春”节气之间出生的小孩，在确定属相的时候就有点麻烦了。属龙还是属蛇？这是个问题。

[算法系列之十九：用天文方法计算日月合朔（新月）](http://blog.csdn.net/orbit/article/details/8223751)

中国农历的朔望月是农历历法的基础，而朔望月又是严格以日月合朔发生的那一天作为月首，因此日月合朔时间的计算是制定农历历法的关键。本文将介绍ELP-2000/82月球运行理论，以及如何用ELP-2000/82月球运行理论计算日月合朔时间。

        要计算日月合朔时间，首先要对日月合朔这一天文现象进行数学定义。朔望月是在地球上观察到的月相周期，平均长度约等于29.53059日，而恒星月（天文月）是月亮绕地球公转一周的时间，长度约27.32166日。月相周期长度比恒星月长大约两天，这是因为在月球绕地球旋转一周的同时，地球还带着它绕太阳旋转了一定的角度的缘故，所以月相周期不仅与月球运行有关，还和太阳运行有关。日月合朔的时候，太阳、月亮和地球三者接近一条直线，月亮未被照亮的一面对着地球，因此地球上看不到月亮，此时又被称为新月。图（1）就是日月合朔天文现象的示意图：



图（1）日月天文现象示意图

月亮绕太阳公转的白道面和地球绕太阳公转的黄道面存在一个最大约5°的夹角，因此大多数情况下，日月合朔时都不是严格在同一条直线上，不过也会发生在同一直线的情况，此时就会发生日食。图（1－b）显示了日月合朔时侧切面上月亮的三种可能的位置情况，当月亮处在位置2时就会发生日食。由图（1）可知，日月合朔的数学定义就是太阳和月亮的地心视黄经差为0的时刻。

        要计算日月合朔，需要知道太阳地心视黄经和月亮地心视黄经的计算方法。“日历生成算法”系列的第三篇《用天文方法计算二十四节气》一文已经介绍了如何用VSOP82/87行星理论计算太阳的地心视黄经，本文将继续介绍如何用ELP-2000/82月球理论计算月亮的地心视黄经。ELP-2000/82月球理论是M. Chapront-Touze和J. Chapront在1983年提出的一个月球位置的半解析理论，和其它半解析理论一样，ELP-2000/82理论也包含一套计算方法和相应的迭代周期项。这套理论共包含37862个周期项，其中20560个用于计算月球经度，7684个用于计算月球纬度，9618个用于计算地月距离。但是这些周期项中有很多都是非常小的值，例如一些计算经纬度的项对结果的增益只有0.00001角秒，还有一些地月距离周期项对距离结果的增益只有0.02米，对于精度不高的历法计算，完全可以忽略。

        有很多基于ELP-2000/82月球理论的改进或简化理论，《天文算法》一书的第四十五章就介绍了一种改进算法，其周期项参数都是从ELP-2000/82理论的周期项参数转换来的，忽略了小的周期项。使用该方法计算的月球黄经精度只有10”，月亮黄纬精度只有4”，但是只用计算60个周期项，速度很快，本文就采用这种修改过的ELP-2000/82理论计算月亮的地心视黄经。这种计算方法的周期项分三部分，分别用来计算月球黄经，月球黄纬和地月距离，三部分的周期项的内容一样，由四个计算辐角的系数和一个正弦（或余弦）振幅组成。计算月球黄经和月球黄纬使用正弦表达式求和：A \* sin(θ)，计算地月距离用余弦表达式求和：A \* cos(θ)，其中辐角θ的计算公式是：

θ = a \* D + b \* M + c \* M’ + d \* F                           (4.1式)

4.1式中的四个辐角系数a、b、c和d由每个迭代周期项给出，日月距角D、太阳平近地角M、月亮平近地角M’以及月球生交点平角距F则分别有4.2式－4.5式进行计算：

D = 297.8502042 + 445267.1115168 \* T - 0.0016300 \* T2

+ T3 / 545868 - T4 / 113065000                   (4.2式)  
M = 357.5291092 + 35999.0502909 \* T - 0.0001536 \* T2

+ T3 / 24490000                                (4.3式)  
M' = 134.9634114 + 477198.8676313 \* T + 0.0089970 \* T2

+ T3 / 69699 - T4 / 14712000                     (4.4式)  
F = 93.2720993 + 483202.0175273 \* T - 0.0034029 \* T2

- T3 / 3526000 + T4 / 863310000                  (4.5式)

以上各式计算结果的单位是度，其中T是儒略世纪数，T计算由4.6式计算：

T = (JDE - 2451545.0) / 36525.0                          (4.6式)

以计算月球黄经的周期项第二项的计算为例，第二项数据如下，辐角系数a = 2，b = 0，c = -1，d = 0，振幅A = 1274027，黄经计算用正弦表达式，则I2的计算如下所示：

I2 = 1274027 \* sin(2D – M’)                            （4.7式）

在套用4.7式计算出60个月球黄经周期项值的时候，需要注意对包含了太阳平近地角M的项进行修正，因为M的值与地球公转轨道的离心率有关，因为离心率是个与时间有关的变量，导致振幅A实际上是个变量，需要根据时间进行修正。月球黄经周期项的修正方法是：如果辐角中包含了M或-M时，需要乘以系数E修正；如果辐角中包含了2M或-2M，则需要乘以系数E的平方进行修正。系数E的计算表达式如下：

E = 1 - 0.002516 \* T - 0.0000074 \* T2                      （4.8式）

其中T值由4.6式计算。上面的计算月球黄经的第二个周期项中M对应的系数是0，因此I2不需要修正，但是第五个周期项中M对应的系数是1，因此I5需要乘以E进行修正。套用4.7式计算出60个月球黄经周期项值I1－I60之和ΣI：

ΣI = I1+ I2 + … + I60                                    （4.9式）

        月球黄纬的周期项和Σb的计算方法与月球黄经周期项和ΣI的计算方法一样，每个月球黄纬周期项也包含振幅A和四个辐角系数a、b、c和d，对于太阳平近地角M的系数b不是0的情况也需要乘以E或E2进行修正。地月距离的周期项和Σr也可以按照上面的方法计算，计算地月距离的目的是为了计算月亮光行差，因为地月距离较小，从地球观察月亮产生的光行差也很小，相对于本文算法的精度（月球黄经精度10”，月亮黄纬精度4”）来说，可以忽略光行差修正，因此就不用计算地月距离。

        由于金星和木星对月球的摄动影响，需要对计算出的月球黄经周期项和ΣI和月球黄纬周期项和Σb金星摄动修正，修正的方法如下：

ΣI += +3958 \* sin( A1 ) + 1962 \* sin( L' - F ) + 318 \* sin( A2 )             （4.10式）

Σb += -2235 \* sin( L' ) + 382 \* sin( A3) + 175 \* sin( A1 - F ) + 175 \* sin( A1 + F )  
       + 127 \* sin( L' - M') - 115 \* sin( L' + M')                           （4.11式）

其中M’和F分别由4.4式和4.5式计算得到，L’是月球平黄经，计算方法是：

L'=218.3164591 + 481267.88134236 \* T - 0.0013268 \* T2

+ T3 / 538841 - T4 / 65194000                         （4.12式）

A1、A2和A4是摄动角修正量，计算方法如下：

A1 = 119.75 + 131.849 \* T                                             （4.13式）  
A2 = 53.09 + 479264.290 \* T                                           （4.14式）  
A3 = 313.45 + 481266.484 \* T                                          （4.15式）

完成所有修正后，就可以用4.16式和4.17式最终得到月亮的地心视黄经和地心视黄纬：

λ = L'+ ΣI / 1000000.0                                               （4.16式）

β = Σb / 1000000.0                                                  （4.17式）

ΣI和Σb最后要除以1000000.0是因为周期项系数中振幅A的单位是0.000001度，因此λ和β的单位是度。下面给出计算月球地心视黄经的代码：

|  |
| --- |
| 123 double GetMoonEclipticLongitudeEC(double dbJD)  124 {  125     double Lp,D,M,Mp,F,E;  126     double dt = (dbJD - JD2000) / 36525.0; /\*儒略世纪数\*/  127  128     GetMoonEclipticParameter(dt, &Lp, &D, &M, &Mp, &F, &E);  129  130     /\*计算月球地心黄经周期项\*/  131     double EI = CalcMoonECLongitudePeriodicTbl(D, M, Mp, F, E);  132  133     /\*修正金星,木星以及地球扁率摄动\*/  134     EI += CalcMoonLongitudePerturbation(dt, Lp, F);  135  136     /\*计算月球地心黄经\*/  137     double longitude = Lp + EI / 1000000.0;  138  139     /\*计算天体章动干扰\*/  140     longitude += CalcEarthLongitudeNutation(dt / 10.0);  141  142     longitude = Mod360Degree(longitude);  143     return longitude;  144 } |

函数参数dbJD是力学时儒略日时间，返回以度为单位的月球视黄经。其中GetMoonEclipticParameter()函数分别根据4.2式、4.3式、4.4式、4.5式、4.8式和4.12式计算日月距角D、太阳平近地角M、月亮平近地角M’、月球生交点平角距F、修正系数E和月球平黄经L’，不需多说明，只要根据以上各式直接计算即可。CalcMoonECLongitudePeriodicTbl()函数计算60个月球黄经周期项和，并根据M值系数的情况进行修正，算法实现如下：

|  |
| --- |
| 42 double CalcMoonECLongitudePeriodicTbl(double D, double M, double Mp, double F,double E)  43 {  44     double EI = 0.0 ;  45  46     for(int i = 0; i < COUNT\_ITEM(Moon\_longitude); i++)  47     {  48         double sita = Moon\_longitude[i].D \* D + Moon\_longitude[i].M \* M +Moon\_longitude[i].Mp \* Mp + Moon\_longitude[i].F \* F;  49         sita = DegreeToRadian(sita);  50         EI += (Moon\_longitude[i].eiA \* sin(sita) \* pow(E,fabs(Moon\_longitude[i].M)));  51     }  52  53     return EI;  54 } |

 CalcMoonLongitudePerturbation()函数计算月球黄经摄动修正量，使用了4.13式和4.14式给出的计算方法：

|  |
| --- |
| 87 double CalcMoonLongitudePerturbation(double dt, double Lp, double F)  88 {  89     double T = dt; /\*T是'ca?从'b4?J2000起'c6?算'cb?的'b5?儒'c8?略'c2?世'ca?纪'bc?数'ca?\*/  90     double A1 = 119.75 + 131.849 \* T;  91     double A2 = 53.09 + 479264.290 \* T;  92  93     A1 = Mod360Degree(A1);  94     A2 = Mod360Degree(A2);  95  96     double result = 3958.0 \* sin(DegreeToRadian(A1));  97     result += (1962.0 \* sin(DegreeToRadian(Lp - F)));  98     result += (318.0 \* sin(DegreeToRadian(A2)));  99  100     return result;  101 } |

        至此，本文已经介绍了使用ELP-2000/82月球理论计算任意时刻月亮地心视黄经的方法，结合“日历生成算法”系列的第三篇《用天文方法计算二十四节气》一文介绍的计算太阳地心视黄经的方法，就可以计算日月合朔的准确时间了。由于ELP-2000/82月球理论也没有根据月球黄经反算时间的方法，因此本文也采用和《用天文方法计算二十四节气》一文中一样的牛顿迭代法计算日月合朔时间。

        关于牛顿迭代法可以参考相关的数学资料，“日历生成算法”系列的第三篇《用天文方法计算二十四节气》一文对如何使用牛顿迭代法有简单的介绍，可以参考一下。总的来说，就是要先定义需要求解的方程f(x)，根据上文的介绍，我们需要求解的是太阳的地心黄经和月亮的地心黄经差值是0的时候的时间，《用天文方法计算二十四节气》一文已经介绍了求太阳地心黄经的函数GetSunEclipticLongitudeECDegree()，本文也给出了求月亮地心黄经的函数GetMoonEclipticLongitudeECDegree()，因此可以定义方程为：

f(x) = GetMoonEclipticLongitudeECDegree(x) – GetSunEclipticLongitudeECDegree(x) = 0

其中x是儒略日单位的，我们要用牛顿迭代法求方程f(x)=0时的解x，也就是时间值。牛顿迭代法求解的迭代式是：

Xn+1 = Xn – f(Xn)/f’(Xn)

这里也不多解释了。导函数仍然使用近似公式，也不解释了，直接上迭代求解的代码了：

|  |
| --- |
| 102 double CalculateMoonShuoJD(double tdJD)  103 {  104     double JD0, JD1,stDegree,stDegreep;  105  106     JD1 = tdJD;  107     do  108     {  109         JD0 = JD1;  110         double moonLongitude = GetMoonEclipticLongitudeECDegree(JD0);  111         double sunLongitude = GetSunEclipticLongitudeECDegree(JD0);  112  113         stDegree = moonLongitude - sunLongitude;  114  115  116         stDegreep = (GetMoonEclipticLongitudeECDegree(JD0 + 0.000005) -GetSunEclipticLongitudeECDegree(JD0 + 0.000005) -GetMoonEclipticLongitudeECDegree(JD0 - 0.000005) +GetSunEclipticLongitudeECDegree(JD0 - 0.000005)) / 0.00001;  117         JD1 = JD0 - stDegree / stDegreep;  118     }while((fabs(JD1 - JD0) > 0.00000001));  119  120     return JD1;  121 } |

        至本文结束，我们已经能够使用半解析算法计算太阳的黄经和月亮的黄经，并且能够通过牛顿迭代法或者24节气的准确时间和日月合朔的准确时间，在这基础上就可以进行中国农历的推算了，“日历生成算法”系列的下一篇将介绍中国农历的历法规则和推算方法。

        再次说明一下，以上算法中讨论的时间都是力学时时间（TD），与国际协调时（UTC）以及各个时区的本地时间都有不同，以上计算出来的时间都需要调整成本地时间，比如中国的中原地区就是东八区标准时（UTC + 8）。应用本文的算法计算出2012年前后15个日月合朔时间如下（已经转换为东八区标准时）：

2011-11-25, 14:09:41.25

2011-12-25, 02:06:27.25

2012-01-23, 15:39:24.16

2012-02-22, 06:34:40.84

2012-03-22, 22:37:08.91

2012-04-21, 15:18:22.12

2012-05-21, 07:46:59.97

2012-06-19, 23:02:06.39

2012-07-19, 12:24:02.83

2012-08-17, 23:54:28.03

2012-09-16, 10:10:36.99

2012-10-15, 20:02:30.98

2012-11-14, 06:08:05.90

2012-12-13, 16:41:37.60

2013-01-12, 03:43:31.34

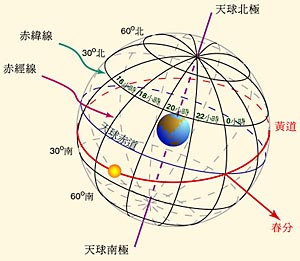
[算法系列之十八：用天文方法计算二十四节气（上）](http://blog.csdn.net/orbit/article/details/7910220)

 二十四节气在中国古代历法中扮演着非常重要的角色，本文将介绍二十四节气的基本知识，以及如何使用VSOP82/87行星运行理论计算二十四节气发生的准确时间。

        中国古代历法都是以月亮运行规律为主，严格按照朔望月长度定义月，但是由于朔望月长度和地球回归年长度无法协调，会导致农历季节和天气的实际冷暖无法对应，因此聪明的古人将月亮运行规律和太阳运行规律相结合制定了中国农历的历法规则。在这种特殊的阴阳结合的历法规则中，二十四节气就扮演着非常重要的作用，它是联系月亮运行规律和太阳运行规律的纽带。正是由于二十四节气结合置闰规则，使得农历的春夏秋冬四季和地球绕太阳运动引起的天气冷暖变化相一致，成为中国几千年来生产、生活的依据。

        二十四节气起源于中国黄河流域。远在春秋时代，古人就开始使用仲春、仲夏、仲秋和仲冬四个节气指导农耕种植。后来经过不断地改进与完善，到秦汉年间，二十四节气已经基本确立。公元前 104年，汉武帝颁布由邓平等人制定的《太初历》，正式把二十四节气订于历法，明确了二十四节气的天文位置。二十四节气天文位置的定义，就是从太阳黄经零度开始，沿黄经每运行15度所经历的时日称为“一个节气”。太阳一个回归年运行360度，共经历24个节气，每个公历月对应2个节气。其中，每月第一个节气为“节令”，即：立春、惊蛰、清明、立夏、芒种、小暑、立秋、白露、寒露、立冬、大雪和小寒等12个节令；每月的第二个节气为“中气”，即：雨水、春分、谷雨、小满、夏至、大暑、处暑、秋分、霜降、小雪、冬至和大寒等12个中气。“节令”和“中气”交替出现，各历时15天，人们习惯上把“节令”和“中气”统称为“节气”。

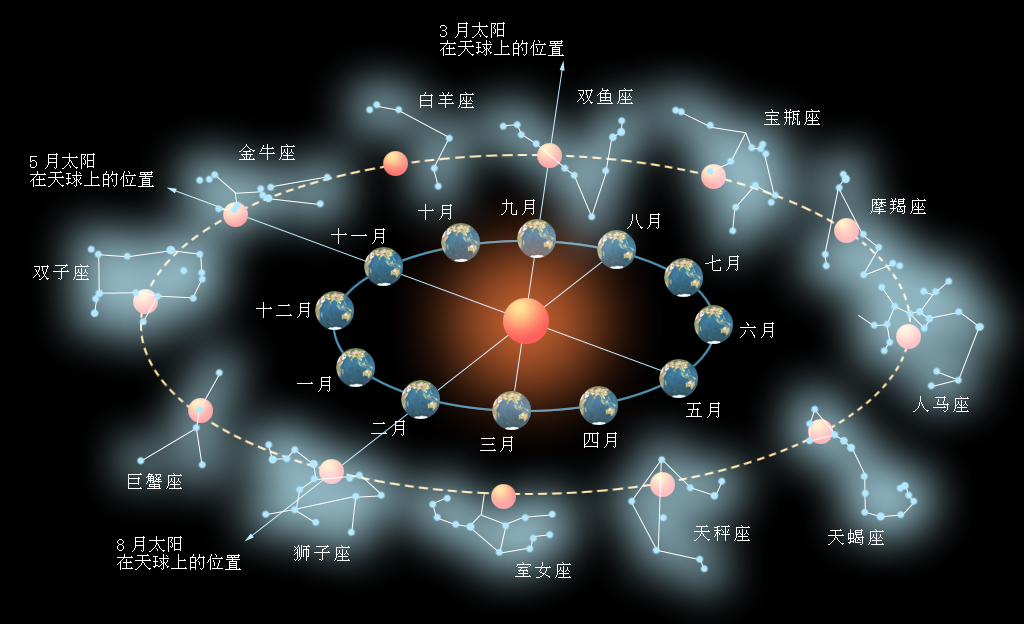
        为了更好地理解二十四节气的天文位置，首先要解释几个天文学概念。“天球”是人们为了研究天体的位置和运动规律而引入的一个假象的球体，根据观察点（也就是球心）的位置不同，可分为“日心天球”、“地心天球”等等。图（1）就是天球概念的一个简单示意图：



图（1）天球概念示意图

天文学中常用的一个坐标体系就是“地心天球”，它与地球同心且有相同的自转轴，理论上具有无限大的半径。地球的赤道和南北极点延伸到天球上，对应着天赤道和南北天极点。和地球上用经纬度定为位置一样，天球也划分了经纬度，分别命名为“赤经”和“赤纬”，地球上的经度用的是度（分秒）为单位，赤经以时（分秒）为单位。天空中的所有天体都可以投射到天球上，用赤经和赤纬定位天体在天球上的位置。“黄道（Ecliptic）”是地球绕太阳公转轨道的轨道平面与天球（地心天球）相交的大圆，由于地球公转受月球和其它行星的摄动，地球的公转轨道并不是严格的平面，因此黄道的严格定义是：地月系质心绕太阳公转的瞬时平均轨道平面与天球相交的大圆。黄道和天赤道所在的两个平面并不是重叠的，它们之间存在一个23度26分的交角，称为“黄赤交角”。由于黄赤交角的存在，黄道和天赤道就在天球上有两个交点，这两个交点就是春分点和秋分点。在天球上以黄道为基圈可以形成黄道坐标系，在黄道坐标系中，也使用了经纬度的概念，分别称为“黄经”和“黄纬”。天体的黄经从春分点起沿黄道向东计量，春分点是黄经0度，沿黄道一周是360度，使用的单位是度、分和秒。黄纬以黄道测量平面为准，向北记为0度到90度，向南记为0度到-90度。

        黄道平面可以近似理解为地球绕太阳公转的平面，以黄道为基圈的黄道坐标系根据观测中心是太阳还是地球还可以区分为日心坐标系和地心坐标系，对应天体的黄道坐标分别被称为“日心黄经、日心黄纬”和“地心黄经、地心黄纬”。日心黄经和日心黄纬比较容易理解，因为太阳系的行星都是绕太阳公转，以太阳为中心将这些行星向天球上投影是最简单的确定行星位置关系的做法。但是人类自古观察太阳的周年运动，都是以地球为参照，以太阳的周年视运动位置来计算太阳的运行轨迹，使用的其实都是地心黄经和地心黄纬，要了解古代历法，理解这一点非常重要。图（2）就解释了造成这种视觉错觉的原因：



图（2）太阳黄道视觉位置原理图

        古人定义二十四节气的位置，是太阳沿着黄道运行时的视觉位置，每个节气对应的黄道经度其实是地心黄经。从图（2）可以看出日心黄经和地心黄经存在180度的转换关系，同样可以理解，日心黄纬和地心黄纬在方向上是反的，因此可以很方便地将两类坐标相互转换，转换公式是：

太阳地心黄经 = 地球日心黄经 + 180°                  （3.1式）

太阳地心黄纬 = -地球日心黄纬                         （3.2式）

【我认为：】

太阳的 地心黄经 = 地球的 日心黄经 + 180°                  （3.1式）

太阳的 地心黄纬 =－地球的 日心黄纬                         （3.2式）

        了解了以上的天文学基础之后，就可以着手对二十四节气的发生时间进行计算。我们常说的节气发生时间，其实就是在太阳沿着黄道做视觉运动过程中，当太阳地心黄经等于某个节气黄经度数时的那个瞬间的时间。所谓的用天文算法计算二十四节气时间，就是根据牛顿力学原理或开普勒三大行星定律，计算出与历法密切相关的地球、太阳和月亮三个天体的运行轨道和时间参数，以此得出当这些天体位于某个位置时的时间。这样的天文计算需要计算者有扎实的微积分学、几何学和球面三角学知识，令广大天文爱好者望而却步。但是随着VSOP-82/87行星理论以及ELP-2000/82月球理论的出现，使得天文计算变得简单易行，本文就是以VSOP-82/87行星理论为计算依据，计算二十四节气的准确时间。

        古代天文学家在对包括地球和月亮在内的行星运行轨道精确计算后发现，天体的运行因为受相近天体的影响，并不严格遵循理论方法计算出来的轨道，而是在理论轨道附近波动。这种影响在天文学上被称为摄动，摄动很难被精确计算，只能根据经验估算。但是经过长期的观测和计算，天文学家发现行星轨道因为摄动影响而产生的波动其实也是有规律的，即在相当长的时间内呈现出周期变化的趋势。于是天文学家开始研究这种周期变化，希望通过一种类似曲线拟合的方法，对一些周期计算项按照某种计算式迭代求和计算代替积分计算来模拟行星运行轨迹。这种计算式可以描述为：a + bt + ct2 + … xcos(p + qt + rt2 + …)，其中t是时间参数，这样的理论通常被称为半解析（semi-analytic）理论。其实早在十八世纪，欧洲学者Joseph Louis Lagrange就开始尝试用这种周期项计算的方法修正行星轨道，但是他采用的周期项计算式是线性方程，精度不高。

        1982年，P.Bretagnon公开发表了VSOP行星理论（这个理论的英文名称是：Secular Variations of the Planetary Orbits，VSOP的缩写其实是源于法文名称：Variations Séculaires des Orbites Planétaires），VSOP理论是一个描述太阳系行星轨道在相当长时间范围内周期变化的半分析（semi-analytic）理论。VSOP82理论是VSOP理论的第一个版本，提供了对太阳系几大行星位置计算的周期序列，通过对周期序列进行正弦或余弦项累加求和，就可以得到这个行星在给定时间的轨道参数。不过VSOP82由于每次都会计算出全部超高精度的轨道参数，这些轨道参数对于历法计算这样的民用场合很不适用。1987年，Bretagnon 和 Francou 创建了VSOP87行星理论，VSOP87行星理论不仅能计算各种精密的轨道参数，还可以直接计算出行星的位置，行星位置可以是各种坐标系，包括黄道坐标系。VSOP87行星理论由6张周期项系数表组成，分别是VSOP87、VSOP87A、VSOP87B、VSOP87C、VSOP87D和VSOP87E，其中VSOP87D表可以直接计算行星日心黄经（L）、日心黄纬（B）和到太阳的距离（R），此表计算出的结果适用于节气位置判断。

        VSOP87D表包含了三部分数据，分别是计算行星日心黄经的周期项系数表（L表）、计算行星日心黄纬的周期项系数表（B表）和计算行星和太阳距离的周期项系数表（R表）。VSOP87D表有太阳系8大行星的数据，本文的计算只关心与地球相关的数据。L表由L0－L5六部分组成，每一部分都包含若干个周期项系数条目，比如L0表有559个周期项系数条目，L1表有341个条目等等。L表的每个周期项系数条目包含若干个参数，用于计算各种轨道参数和位置参数，计算地球的日心黄经只需要用到其中三个系数。计算所有的周期项系数并不是必须的，有时候减少一些系数比较小的周期项可以减少计算所花费的时间，当然，这会牺牲一点精度。假设计算地球日心黄经的三个系数是A、B和C，则每个周期项的计算表达式是：

A \* cos(B + Cτ)                               （3.3式）

其中τ是儒略千年数，τ的计算公式如下：

τ = (JDE - 2451545.0) / 365250                    （3.4式）

JDE是计算轨道参数的时间，单位是儒略日，2451545.0是公元2000年1月1日 12时的儒略日数，关于儒略日的概念，请参考“日历生成算法”的第一篇《中国公历（格里历）》中的说明以及计算方法。以L0表的第二个周期项为例，这个周期项数据中与日心黄经计算有关的三个系数分别是A= 3341656.456，B=4.66925680417，C=6283.07584999140，则第二个周期项的计算方法是：3341656.456 \* cos(4.66925680417 + 6283.0758499914 \* τ)。对L0表的各项分别计算后求和可得到L0表周期项总和L0，对L表的其它几个部分使用相同的方法计算周期项和，可以得到L1、L2、L3、L4和L5，然后用用3.5式计算出最终的地球日心黄经，单位是弧度：

L = (L0 + L1 \* τ+ L2 \* τ2 + L3 \* τ3 + L4 \* τ4 +L5 \* τ5) / 108          （3.5式）

用同样的方法对地球日心黄纬的周期项系数表和计算行星和太阳距离的周期项系数表计算求和，可以得到地球日心黄纬B和日地距离R，B的单位是弧度，R的单位是天文单位（AU）[1]。由于3.5式的计算方法需要多次计算τ的乘方，浮点数的乘方计算的速度比较慢，实际计算时，通常对3.5式进行变换，用乘法和加法代替直接的乘方计算，这是一种常用的转换：

L = (((((L5 \* τ + L4) \* τ + L3) \* τ + L2) \* τ + L1) \* τ + L0) / 108        （3.6式）

本文就是使用3.6式代替3.5式进行计算。

        VSOP82/87行星理论中的周期项系数对不同的行星具有不同的精度，对地球来说，在1900-2100年之间的200年跨度期间，计算精度是0.005"。前文曾说过，对于不需要这么高精度的计算应用时，可以适当减少一些系数比较小的周期项，减少计算量，提高计算速度。Jean Meeus在他的《天文算法》一书中就给出了一套精简后的VSOP87D表的周期项，将计算地球黄经的L0表由原来的559项精简到64项，计算地球黄纬的B0表甚至被精简到只有5项，从实际效果看，计算精度下降并不多，但是极大地减少了计算量。

        使用VSOP87D周期项系数表计算得到的是J2000.0平黄道和平春分点（mean dynamic ecliptic and equinox）为基准的日心黄经（L）和日心黄纬（B），其值与标准FK5系统略有差别，如果对精度要求很高可以采用下面的方法将计算得到的日心黄经（L）和日心黄纬（B）转到FK5系统[2]：

首先然后 L'，单位是度：

 L' = L - 1.397 \* T - 0.00031\*T2                                （3.7式）

3.7式中的T是儒略世纪数，它与儒略千年数τ的关系是：T = 10 \*τ。然后使用L'计算L和B的修正值ΔL和ΔB：

 ΔL = -0.09033 + 0.03916 \* ( cos(L') + sin(L') ) \* tan(B)             （3.8式）

 ΔB = +0.03916 \* ( cos(L') - sin(L') )                             （3.9式）

 ΔL和ΔB的单位都是"，是度分秒角度单位体系，需要将3.6式计算出得L和B转换成以度（°）为单位的值后再进行修正。

        CalcSunEclipticLongitudeEC()函数就是使用VSOP87行星理论计算行星日心黄经的代码实现，整个计算过程和前文描述一样，首先根据VSOP87D表的数据计算出L0-L5，然后用3.6式计算出地球的日心黄经，3.6式计算出来的单位是弧度，因此转换成度分秒单位，最后使用3.1式将结果转换成太阳的地心黄经：

|  |
| --- |
| double CalcSunEclipticLongitudeEC(double dt)   {       double L0 = CalcPeriodicTerm(Earth\_L0, sizeof(Earth\_L0) /sizeof(VSOP87\_COEFFICIENT), dt);       double L1 = CalcPeriodicTerm(Earth\_L1, sizeof(Earth\_L1) /sizeof(VSOP87\_COEFFICIENT), dt);       double L2 = CalcPeriodicTerm(Earth\_L2, sizeof(Earth\_L2) /sizeof(VSOP87\_COEFFICIENT), dt);       double L3 = CalcPeriodicTerm(Earth\_L3, sizeof(Earth\_L3) /sizeof(VSOP87\_COEFFICIENT), dt);       double L4 = CalcPeriodicTerm(Earth\_L4, sizeof(Earth\_L4) /sizeof(VSOP87\_COEFFICIENT), dt);       double L5 = CalcPeriodicTerm(Earth\_L5, sizeof(Earth\_L5) /sizeof(VSOP87\_COEFFICIENT), dt);         double L = (((((L5 \* dt + L4) \* dt + L3) \* dt + L2) \* dt + L1) \* dt + L0) /100000000.0;         /\*地心黄经 = 日心黄经 + 180度\*/       return (Mod360Degree(Mod360Degree(L / RADIAN\_PER\_ANGLE) + 180.0));   } |

Mod360Degree()函数将大于360°或小于0°的值调整到0－360°之间，便于转换显示。CalcPeriodicTerm()函数使用3.3式对一个周期项系数表进行求和计算，可以指定需要计算的周期项数：

|  |
| --- |
| double CalcPeriodicTerm(const VSOP87\_COEFFICIENT \*coff, int count, double dt)   {       double val = 0.0;         for(int i = 0; i < count; i++)           val += (coff[i].A \* cos((coff[i].B + coff[i].C \* dt)));         return val;   } |

同样的方法计算太阳的地心黄纬：

|  |
| --- |
| double CalcSunEclipticLatitudeEC(double dt)   {       double B0 = CalcPeriodicTerm(Earth\_B0, sizeof(Earth\_B0) /sizeof(VSOP87\_COEFFICIENT), dt);       double B1 = CalcPeriodicTerm(Earth\_B1, sizeof(Earth\_B1) /sizeof(VSOP87\_COEFFICIENT), dt);       double B2 = CalcPeriodicTerm(Earth\_B2, sizeof(Earth\_B2) /sizeof(VSOP87\_COEFFICIENT), dt);       double B3 = CalcPeriodicTerm(Earth\_B3, sizeof(Earth\_B3) /sizeof(VSOP87\_COEFFICIENT), dt);       double B4 = CalcPeriodicTerm(Earth\_B4, sizeof(Earth\_B4) /sizeof(VSOP87\_COEFFICIENT), dt);         double B = (((((B4 \* dt) + B3) \* dt + B2) \* dt + B1) \* dt + B0) / 100000000.0;        /\*地心黄纬 = －日心黄纬\*/      return -(B / RADIAN\_PER\_ANGLE);   } |

AdjustSunEclipticLongitudeEC()函数根据3.8式计算黄经的修正量，longitude和latitude参数是由VSOP87理论计算出的太阳地心黄经和地心黄纬，单位是度，dt是儒略千年数，返回值单位是度：

|  |
| --- |
| double AdjustSunEclipticLongitudeEC(double dt, double longitude, double latitude)   {       double T = dt \* 10; //T是儒略世纪数         double dbLdash = longitude - 1.397 \* T - 0.00031 \* T \* T;         // 转换为弧度       dbLdash \*= RADIAN\_PER\_ANGLE;         return (-0.09033 + 0.03916 \* (cos(dbLdash) + sin(dbLdash)) \* tan(latitude \*RADIAN\_PER\_ANGLE)) / 3600.0;   } |

        经过上述计算转换得到坐标值是理论值，或者说是天体的几何位置，但是FK5系统是一个目视系统，也就是说体现的是人眼睛观察效果（光学位置），这就需要根据地球的物理环境、大气环境等信息做进一步的修正，使其和人类从地球上观察星体的观测结果一致。

【下篇将介绍修正理论和修正算法，请继续关注】

**小知识1：天文单位**

        天文单位（英文：Astronomical Unit，简写AU）是一个长度的单位，约等于地球跟太阳的平均距离。天文单位是天文常数之一，是天文学中测量距离，特别是测量太阳系内天体之间的距离的基本单位。地球到太阳的平均距离大约为一个天文单位，约等于1.496亿千米。 1976年，国际天文学联会把一天文单位定义为一颗质量可忽略、公转轨道不受干扰而且公转周期为365.2568983日（即一高斯年）的粒子与一个质量相等约一个太阳的物体的距离。当前普遍被接受并使用的天文单位的值是149,597,870,691±30米（约一亿五千万公里）。

**小知识2：FK5系统**

        FK5常用的目视星表系统，又称第五基本星表，是在FK4（第四基本星表）的基础上发展出来的，对FK4星表进行了修正，于1984年正式启用。它定义了一个以太阳质心为中心，J2000.0平赤道和春分点为基准的天球平赤道坐标系。近年来国际上又编制了FK6星表（第六基本星表），但是还没有被正式启用。

[算法系列之十八：用天文方法计算二十四节气（下）](http://blog.csdn.net/orbit/article/details/7944248)

【接上篇】

        经过上述计算转换得到坐标值是理论值，或者说是天体的几何位置，但是FK5系统是一个目视系统，也就是说体现的是人眼睛观察效果（光学位置），这就需要根据地球的物理环境、大气环境等信息做进一步的修正，使其和人类从地球上观察星体的观测结果一致。

        首先需要进行章动修正。章动是指地球沿自转轴的指向绕黄道极缓慢旋转过程中，由于地球上物质分布不均匀性和月球及其它行星的摄动力造成的轻微抖动。英国天文学家詹姆斯·布拉德利(1693—1762)最早发现了章动，章动可以沿着黄道分解为水平分量和垂直分量，黄道上的水平分量记为Δψ，称为黄经章动，它影响了天球上所有天体的经度。黄道上的垂直分量记为Δε，称为交角章动，它影响了黄赤交角。目前编制天文年历所依据的章动理论是伍拉德在1953年建立的，它是以刚体地球模型为基础的。1977年，国际天文联合会的一个专家小组建议采用非刚体地球模型――莫洛坚斯基II模型代替刚体地球模型计算章动，1979年的国际天文学联合会第十七届大会正式通过了这一建议，并决定于1984年正式实施。

        地球章动主要是月球运动引起的，也具有一定的周期性，可以描述为一些周期项的和，主要项的周期是6798.4日(18.6年)，但其它项是一些短周期项(小于10天)。本文采用的计算方法取自国际天文联合会的IAU1980章动理论，周期项系数数据来源于《天文算法》一书第21章的表21－A，该表忽略了IAU1980章动理论中系数小于0.0003"的周期项，因此只有63项。每个周期项包括计算黄经章动（Δψ）的正弦系数（相位内项系数）、计算交角章动的（Δε）余弦系数（相位外项系数）以及计算辐角的5个基本角距（M、M'、D、F、Ω）的线性组合系数。5个基本角距的计算公式是：

平距角(日月对地心的角距离)：  
D = 297.85036 + 455267.111480 \* T - 0.0019142 \* T2 + T3 / 189474        （3.10式）  
太阳（地球）平近点角：  
M = 357.52772 + 35999.050340 \* T - 0.0001603 \* T2 - T3 / 300000         （3.11式）  
月球平近点角  
M'= 134.96298 + 477198.867398 \* T + 0.0086972 \* T2 + T3 / 56250        （3.12式）

月球纬度参数：  
F = 93.27191 + 483202.017538 \* T - 0.0036825 \* T2 + T3 / 327270          （3.13式）  
黄道与月球平轨道升交点黄经：  
Ω= 125.04452 - 1934.136261 \* T + 0.0020708 \* T2 + T3 / 450000            （3.14式）

以上各式中的T是儒略世纪数，计算出来的5个基本角距的单位都是度，在计算正弦或余弦时要转换为弧度单位。计算每一个周期项的黄经章动过程是这样的，首先将3.10－3.14式计算出来的值与对应的5个基本角距系数组合，计算出辐角。以本文使用的章动周期项系数表中的第七项为例，5个基本角距对应的系数分别是1、0、-2、2和2，辐角θ的值就是：-2D + M + 2F + 2Ω。计算出辐角后就可以计算周期项的值：

S = (S1+ S2 \* T) \* sin(θ)                          （3.15式）

仍以第七项为例，S的值就是（-517 + 1.2 \* T）\* sin(θ)。对每一项的值S累加就可得到黄经章动，单位是0.0001"。交角章动的计算方法与黄经章动的计算类似，辐角θ的值是一样的，只是计算章动使用的是余弦系数：

C = (C1 + C2 \* T) \* cos(θ)                          （3.16式）

CalcEarthLongitudeNutation()函数就是计算黄经章动的实现代码：

|  |
| --- |
| double CalcEarthLongitudeNutation(double dt)   {       double T = dt \* 10;       double D,M,Mp,F,Omega;         GetEarthNutationParameter(dt, &D, &M, &Mp, &F, &Omega);         double resulte = 0.0 ;       for(int i = 0; i < sizeof(nutation) / sizeof(nutation[0]); i++)       {           double sita = nutation[i].D \* D + nutation[i].M \* M + nutation[i].Mp \* Mp +nutation[i].F \* F + nutation[i].omega \* Omega;             resulte += (nutation[i].sine1 + nutation[i].sine2 \* T ) \* sin(sita);       }         /\*先乘以章动表的系数 0.0001，然后换算成度的单位\*/       return resulte \* 0.0001 / 3600.0;   } |

 GetEarthNutationParameter()辅助函数用于计算5个基本角距：

|  |
| --- |
| void GetEarthNutationParameter(double dt, double \*D, double \*M, double \*Mp, double\*F, double \*Omega)   {       double T = dt \* 10; /\*T是从J2000起算的儒略世纪数\*/       double T2 = T \* T;       double T3 = T2 \* T;         /\*平距角（如月对地心的角距离）\*/       \*D = 297.85036 + 445267.111480 \* T - 0.0019142 \* T2 + T3 / 189474.0;         /\*太阳（地球）平近点角\*/       \*M = 357.52772 + 35999.050340 \* T - 0.0001603 \* T2 - T3 / 300000.0;         /\*月亮平近点角\*/       \*Mp = 134.96298 + 477198.867398 \* T + 0.0086972 \* T2 + T3 / 56250.0;         /\*月亮纬度参数\*/       \*F = 93.27191 + 483202.017538 \* T - 0.0036825 \* T2 + T3 / 327270.0;         /\*黄道与月亮平轨道升交点黄经\*/       \*Omega = 125.04452 - 1934.136261 \* T + 0.0020708 \* T2 + T3 / 450000.0;   } |

同样，计算交角章动的实现代码是：

|  |
| --- |
| double CalcEarthObliquityNutation(double dt)   {       double T = dt \* 10; /\*T是从J2000起算的儒略世纪数\*/       double D,M,Mp,F,Omega;         GetEarthNutationParameter(dt, &D, &M, &Mp, &F, &Omega);         double resulte = 0.0 ;       for(int i = 0; i < sizeof(nutation) / sizeof(nutation[0]); i++)       {           double sita = nutation[i].D \* D + nutation[i].M \* M + nutation[i].Mp \* Mp +nutation[i].F \* F + nutation[i].omega \* Omega;             resulte += (nutation[i].cosine1 + nutation[i].cosine2 \* T ) \* cos(sita);       }         /\*先乘以章动表的系数 0.001，然后换算成度的单位\*/       return resulte \* 0.0001 / 3600.0;   } |

         除了章动修正，对于目测系统来说，还要进行光行差修正。光行差是指在同一瞬间，运动中的观察者所观测到的天体视方向与静止的观测者所观测到天体的真方向之差。造成光行差的原因有两个，一个是光的有限速度，另一个是观察者的运动。在地球上的天文观测者因和地球一起运动（自传＋公转），他所看到的星光方向与假设地球不动时看到的方向不一样。以太阳为例，光线从太阳传到地球需要约8分钟的时间，在这8分钟多的时间中，地球沿着公转轨道移动了一段距离人们根据现在的观察认定太阳在那个视位置，事实上那是8分钟前太阳的位置。在精确的天文计算中，需要考虑这种光行差引起的视位置差异，在计算太阳的地心视黄经时，要对其进行光行差修正。地球上的观测者可能会遇到几种光行差，分别是因地球公转引起的周年光行差，因地球自传引起的周日光行差，还有因太阳系或银河系运动形成的长期光行差等等，对于从地球上观察太阳这种情况，只需要考虑周年光行差和周日光行差。因太阳公转速度比较快，周年光行差最大可达到20.5角秒，在计算太阳视黄经时需要考虑修正。地球自传速度比较慢，周日光行差最大约为零点几个角秒，因此计算太阳视黄经时忽略周日光行差。

        下面是一个粗略计算太阳地心黄经光行差修正量的公式，其中R是地球和太阳的距离：

AC = -20".4898 / R                                    （3.17式）

分子20.4898并不是一个常数，但是其只的变化非常缓慢，在0年是20".4893，在4000年是20".4904。前文提到过，太阳到地球的距离R可以用VSOP87D表的R0－R5周期项计算出来，R的单位是“天文单位（AU）”，和计算太阳地心黄经和地心黄纬类似，太阳到地球的距离可以这样算出来：

|  |
| --- |
| double CalcSunEarthRadius(double dt)   {       double R0 = CalcPeriodicTerm(Earth\_R0, sizeof(Earth\_R0) /sizeof(VSOP87\_COEFFICIENT), dt);       double R1 = CalcPeriodicTerm(Earth\_R1, sizeof(Earth\_R1) /sizeof(VSOP87\_COEFFICIENT), dt);       double R2 = CalcPeriodicTerm(Earth\_R2, sizeof(Earth\_R2) /sizeof(VSOP87\_COEFFICIENT), dt);       double R3 = CalcPeriodicTerm(Earth\_R3, sizeof(Earth\_R3) /sizeof(VSOP87\_COEFFICIENT), dt);       double R4 = CalcPeriodicTerm(Earth\_R4, sizeof(Earth\_R4) /sizeof(VSOP87\_COEFFICIENT), dt);         double R = (((((R4 \* dt) + R3) \* dt + R2) \* dt + R1) \* dt + R0) / 100000000.0;         return R;   } |

也可以不使用VSOP，而用下面的公式直接计算日地距离R：

R = 1.000001018 (1 - e2) / (1 + e \* cos(v))                 （3.18式）

其中e是地球轨道的离心率：

e = 0.016708617 - 0.000042037 \* T - 0.0000001236 \* T2      （3.19式）

v的计算公式是v = M + C，其中M是太阳平近地角：

M = 357.52910 + 35999.05030 \* T - 0.0001559 \* T2 - 0.00000048 \* T3    （3.20式）

中心C的太阳方程：

C = (1.914600 - 0.004817 \* T - 0.000014 \* T2) \* sin(M)

+ (0.019993 - 0.000101 \* T) \* sin(2M)

+ 0.000290 \* sin(3M)                                              （3.21式）

以上各式中的T都是儒略世纪数，M和C的单位都是度，带入3.18式计算时需要转换成弧度单位，计算出R以后，就可以这样计算光行差修正量：

AC = K / R (K是光行差常数，K = 20".49552)          （3.22式）

无论是使用3.17式还是使用3.22式，最终计算出来的太阳光行差修正单位都是角秒。

        由VSOP87理论计算出来的几何位置黄经，经过坐标转换，章动修正和光行差修正后，就可以得到比较准确的太阳地心视黄经，GetSunEclipticLongitudeEC()函数就是整个过程的代码：

|  |
| --- |
| double GetSunEclipticLongitudeEC(double jde)   {       double dt = (jde - JD2000) / 365250.0; /\*儒略千年数\*/         // 计算太阳的地心黄经       double longitude = CalcSunEclipticLongitudeEC(dt);         // 计算太阳的地心黄纬       double latitude = CalcSunEclipticLatitudeEC(dt) \* 3600.0;         // 修正精度       longitude += AdjustSunEclipticLongitudeEC(dt, longitude, latitude);         // 修正天体章动       longitude += CalcEarthLongitudeNutation(dt);         // 修正光行差       /\*太阳地心黄经光行差修正项是: -20".4898/R\*/       longitude -= (20.4898 / CalcSunEarthRadius(dt)) / (20 \* PI);         return longitude;   } |

参数jde是力学时时间，单位是儒略日，返回太阳地心视黄经，单位是度。

        到现在为止，我们已经知道如何使用VSOP82/87理论计算以儒略日为单位的任意时刻的太阳地心视黄经，但是这和实际历法计算需求还不一致，历法计算需要根据太阳地心视黄经反求出此时的时间。VSOP82/87理论没有提供反向计算的方法，但是可以采用根据时间正向计算太阳视黄经，配合误差修正进行迭代计算的方法，使正向计算出来的结果向已知结果收敛，当达到一定的迭代次数或计算结果与已知结果误差满足精度要求时，停止迭代，此时的正向输入时间就是所求的时间。地球公转轨道是近似椭圆轨道，轨道方程不具备单调性，但是在某个节气附件的一小段时间区间中，轨道方程具有单调性，这个是本文迭代算法的基础。

        实际上，我们要做的事情就是求解方程的根，但是我们面临的这个方程没有求根公式。对此类问题，数学上通常采用的迭代求解方法有二分逼近法和牛顿迭代法，事实上二分逼近法可以用更好的策略，比如用黄金分割代替二分法进行逼近区间的选择。接下来我们将分别介绍这两种方法在计算二十四节气中的应用，首先介绍黄金分割逼近法。

        已知太阳视黄经的值，反求对应的时间的过程是这样的，首先根据节气对应的视黄经角度值W，估算出节气可能的时间区间[A, B]，然后找到这个时间区间内黄金分割点对应的时间值C，C的计算采用3.23式：

C = ((B - A) \* 0.618) + A                               (3.23式)

用C值估算出太阳视黄经W’，如果W’ > W，则调整调迭代时间区间为[A, C]，如果W’ < W，则调整迭代时间区间为[C, B]，然后重复上述过程，直到W’ 与W的差值满足精度要求为止（区间上下限A和B的差值小于门限制也可以作为迭代退出条件）。采用黄金分割法进行逼近求值的算法实现如下：

|  |
| --- |
| 34 double CalculateSolarTerms(int year, int angle)  35 {  36     double lJD, rJD;  37     EstimateSTtimeScope(year, angle, lJD, rJD); /\*估算迭代起始时间区间\*/  38  39     double solarTermsJD = 0.0;  40     double longitude = 0.0;  41  42     do  43     {  44         solarTermsJD = ((rJD - lJD) \* 0.618) + lJD;  45         longitude = GetSunEclipticLongitudeECDegree(solarTermsJD);  46         /\*  47             对黄经0度迭代逼近时，由于角度360度圆周性，估算黄经值可能在(345,360]和[0,15)两个区间，  48             如果值落入前一个区间，需要进行修正  49         \*/  50         longitude = ((angle == 0) && (longitude > 345.0)) ? longitude - 360.0 :longitude;  51  52         (longitude > double(angle)) ? rJD = solarTermsJD : lJD = solarTermsJD;  53     }while((rJD - lJD) > 0.0000001);  54  55     return solarTermsJD;  56 } |

这里要特别说明一下，由于角度的360度圆周性，当在太阳黄经0度附近逼近时，区间的上下界可能分别位于（345, 360]和[0, 15)两个区间上，此时需要将（345, 360]区间修正为（-15, 0]，使得逼近区间边界的选取能够正常进行。EstimateSTtimeScope()函数估算节气的时间区间，估算的依据是每个月的节气时间比较固定，最多相差一两天,考虑的几千年后岁差的影响，这个估算范围还可以再放宽一点，比如，对于月内的第一个节气，可以将时间范围估算为4日到9日，对于月内的第二个节气，可以将时间范围估算为16日到24日，保证迭代范围内有解。EstimateSTtimeScope()函数算法简单，这里就不列出代码了。

        二分逼近或黄金分割逼近算法实现简单，很容易控制，但是也存在效率低，收敛速度慢的问题，现在我们介绍牛顿迭代法，牛顿迭代法是一种在实数域和复数域上近似求解方程的方法。假设我们要求解的方程是f(x) = 0，如果f(x)的导函数f’(x)是连续的，则在真实解x附近的区域内任意一点x0开始迭代，则牛顿迭代法必收敛，特别当f’(x)不等于0的时候，牛顿迭代法是平方收敛的，也就是说，每迭代一次，结果的有效数字将增加一倍。

        简单的说，对于方程f(x) = 0，f(x)的导函数是f’(x)，则牛顿迭代法的迭代公式是：

Xn+1 = xn – f(xn)/f’(xn)                              (3.24式)

现在问题就是如何确定方程f(x)。对于我们面临的问题，可以理解为已知angle，通过GetSunEclipticLongitudeEC(solarTermsJD)函数反向求解solarTermsJD的值，因此我们的方程可以理解为：

f(x) = GetSunEclipticLongitudeEC(x) – angle = 0

确定了方程f(x)，剩下的问题就是求导函数f’(x)。严格的求解，应该根据GetSunEclipticLongitudeEC()函数，以儒略千年数dt为自变量，按照函数求导的规则求出导函数。因为GetSunEclipticLongitudeEC()函数内部是调用其他函数，因此可以理解为是一个多个函数组合的复合函数，类似f(x) = g(x) + h(x, k(x)) + p(x)这样的形式，可以按照求导规则逐步对其求导得到导函数。但是我不打算这么做，因为我有更简单的方法，那就是使用计算导数的近似公式。其实求导函数的目的就是为了得到某一点的导数，如果有近似公式可以直接得到这一点的导数，就不用费劲求导函数了。

        如果函数f(x)是单调函数，或者是在某个区间上是单调函数，则在此函数的其单调区间上某一点的导数值可以用近似公式计算，这个近似公式是：

f’(x0) = (f(x0 + 0.000005) – f(x0 – 0.000005)) / 0.00001            (3.25式)

这是一个精度很高的近似公式，完全可以满足民用历法计算的精度要求。

        根据以上分析结果，使用牛顿迭代法求解节气的算法就很容易实现了，以下就是牛顿迭代法求解节气的代码：

|  |
| --- |
| 74 double CalculateSolarTermsNewton(int year, int angle)  75 {  76     double JD0, JD1,stDegree,stDegreep;  77  78     JD1 = GetInitialEstimateSolarTerms(year, angle);  79     do  80     {  81         JD0 = JD1;  82         stDegree = GetSunEclipticLongitudeECDegree(JD0) - angle;  83         stDegreep = (GetSunEclipticLongitudeECDegree(JD0 + 0.000005)  84                       - GetSunEclipticLongitudeECDegree(JD0 - 0.000005)) /0.00001;  85         JD1 = JD0 - stDegree / stDegreep;  86     }while((fabs(JD1 - JD0) > 0.0000001));  87  88     return JD1;  89 } |

经过验证，牛顿迭代法具有非常好的收敛效果，一般只需3次迭代就可以得到满足精度的结果。

        至此，我们就有了完整的计算节气发生时间的方法，输入年份和节气对应的太阳黄经度数，即可求的节气发生的精确时间。最后说明一下，以上算法中讨论的时间都是力学时时间（TD），与国际协调时（UTC）以及各个时区的本地时间都有不同，以上计算出来的时间都需要调整成本地时间，比如中国的中原地区就是东八区标准时（UTC + 8）。关于力学时、国际协调时（世界时）的定义，请参考文末的小知识3：力学时、原子时和国际协调时。应用本文的算法计算出2012年各个节气的时间如下（已经转换为东八区标准时），与紫金山天文台发布的《2012中国天文年历》中发布的时间在分钟级别上完全吻合（此年历只精确到分钟）：

2012-01-06, 06:43:54.28   小寒

2012-01-21, 00:09:49.08   大寒

2012-02-04, 18:22:22.53   立春

2012-02-19, 14:17:35.37   雨水

2012-03-05, 12:21:01.56   惊蛰

2012-03-20, 13:14:24.17   春分

2012-04-04, 17:05:34.65   清明

2012-04-20, 00:12:03.28   谷雨

2012-05-05, 10:19:39.54   立夏

2012-05-20, 23:15:30.28   小满

2012-06-05, 14:25:52.96   芒种

2012-06-21, 07:08:46.98   夏至

2012-07-07, 00:40:42.66   小暑

2012-07-22, 18:00:50.72   大暑

2012-08-07, 10:30:31.88   立秋

2012-08-23, 01:06:48.41   处暑

2012-09-07, 13:28:59.41   白露

2012-09-22, 22:48:57.14   秋分

2012-10-08, 05:11:41.45   寒露

2012-10-23, 08:13:32.83   霜降

2012-11-07, 08:25:56.47   立冬

2012-11-22, 05:50:08.09   小雪

2012-12-07, 01:18:55.23   大雪

2012-12-21, 19:11:35.61   冬至

**小知识3：力学时、原子时和国际协调时**

        力学时全称是“牛顿力学时”，也被称作是“历书时”。它描述天体运动的动力学方程中作为时间自变量所体现的时间，或天体历表中应用的时间，是由天体力学的定律确定的均匀时间。力学时的初始历元取为1900年初附近，太阳几何平黄经为279°41′48″.04的瞬间，秒长定义为1900.0年回归年长度的1／31556925.9747。1958年国际天文学联合会决议决定：自1960年开始用力学时代替世界时作为基本的时间计量系统，规定天文年历中太阳系天体的位置都按力学时推算。力学时与世界时之差由观测太阳系天体（主要是月球）定出，因此力学时的测定精度较低，1967年起被原子时代替作为基本时间计量系统。

        国际协调时又称世界时，是以本初子午线的平子夜起算的平太阳时，又称格林威治时间。世界各地地方时与世界时之差等于该地的地理经度。世界时1960年以前曾作为基本时间计量系统被广泛应用。由于地球自转速度变化的影响，它不是一种均匀的时间系统。后来世界时先后被历书时和原子时所取代。

        原子时是以物质的原子内部发射的电磁振荡频率为基准的时间计量系统。原子时的初始历元规定为1958年1月1日世界时0时，秒长定义为铯-133原子基态的两个超精细能级间在零磁场下跃迁辐射9192631770周所持续的时间。这是一种均匀的时间计量系统。1967年起，原子时已取代力学时作为基本时间计量系统。

**参考文章：**

[1] 《Secular variations of the planetary orbits》http://www.worldlingo.com/ma/enwiki/en/Secular\_variations\_of\_the\_planetary\_orbits

[2] Jean.Meeus.Astronomical.Algorithms(天文算法)

[3] M.Chapront-Touze and J.Chapront.ELP 2000-85 - A semi-analytical lunar ephemeris adequate for historical times.Astronomy And Astrophysics,1998

[4] P.Bretagnon and G.Francou.Planetray theories in rectangular and spherical variables VSOP87 solutions. Astronomy And Astrophysics,1998

[算法系列之十七：日历生成算法-中国公历（格里历）（上）](http://blog.csdn.net/orbit/article/details/7749723)

日历在我们的生活中扮演着十分重要的角色，上班、上学、约会都离不开日历。每年新年开始，人们都要更换新的日历，你想知道未来一年的这么多天是怎么被确定下来的吗？为什么去年的国庆节是星期五而今年的国庆节是星期三？那就来研究一下日历算法吧。本文将介绍日历的编排规则，确定某日是星期几的计算方法，以及如何在计算机上打印某一年的年历。

        要研究日历算法，首先要知道日历的编排规则，也就是历法。所谓历法，指的就是推算年、月、日的时间长度和它们之间的关系，指定时间序列的法则。我国的官方历法是中国公历，也就是世界通用的格里历（Gregorian Calendar），中国公历的年分为平常年和闰年，平常年一年是365天，闰年一年是366天。判定一年是平常年还是闰年的规则如下：

1、  如果年份是4的倍数，且不是100的倍数，则是闰年；

2、  如果年份是400的倍数，则是闰年；

3、  不满足1、2条件的就是平常年。

总结成一句话就是：四年一闰，百年不闰，四百年再闰。

        中国公历关于月的规则是这样的，一年分为十二个月，其中一月、三月、五月、七月、八月、十月和十二月是大月，一个月有31天。四月、六月、九月和十一月是小月，一个月有30天。二月天数要根据是否是闰年来定，如果是闰年，二月是29天，如果是平常年，二月是28天。

        除了年月日，人们日常生活中还对日期定义了另一个属性，就是星期几。星期并不是公历范畴内的东西，但是人们已经习惯用星期来管理和规划时间，比如一个星期工作五天，休息两天等等，星期的规则彻底改变了人们的生活习惯，因此星期已经成为历法中的一部分了。星期的命名最早起源于古巴比伦文化。公元前7－6世纪，巴比伦人就使用了星期制，一个星期中的每一天都有一个天神掌管。这一规则后来传到古罗马，并逐渐演变成现在的星期制度。

        如何知道某一天到底是星期几？除了查日历之外，是否有办法推算出来某一天是星期几呢？答案是肯定的，星期不象年和月那样有固定的历法规则，但是星期的计算也有自己的规律。星期是固定的7天周期，其排列顺序固定，不随闰年、平常年以及大小月的天数变化影响。因此，只要确切地知道某一天是星期几，就可以推算出其它日期是星期几。推算的方法很简单，就是计算两个日期之间相差多少天，用相差的天数对7取余数，这个余数就是两个日期的星期数的差值。举个例子，假设已经知道1977年3月27日是星期日，如何得知1978年3月27日是星期几？按照前面的方法，计算出1977年3月27日到1978年3月27日之间相差365天，365除以7余数是1，所以1978年3月27日就是星期一。

        上述方法计算星期几的关键是求出两个日期之间相隔的天数。有两种常用的方法计算两个日期之间相隔的天数，一种是利用公历的月和年的规则直接计算，另一种是利用儒略日计算。利用公历规则直接计算两个日期之间相差的天数，简单地讲就是将两个日期之间相隔的天数分成三个部分：前一个日期所在年份还剩下的天数、两个日期之间相隔的整数年所包含的天数和后一个日期所在的年过去的天数。如果两个日期是相邻两个年份的日期，则第二部分整年的天数就是0。以1977年3月27日到2005年5月31日为例，1977年还剩下的天数是279天，中间整数年是从1978年到2005年（不包括2005年），共26年，包括7个闰年和20个平常年，总计9862天，最后是2005年从1月1日到5月31日经过的天数151天。三者总结10292天。直接利用公历规则计算日期相差天数的算法实现如下（为了简化算法复杂度，这个实现假设用于定位星期的那个日期总是在需要计算星期几的那个日期之前）：

|  |
| --- |
| 99 int CalculateDays(int ys, int ms, int ds, int ye, int me, int de)  100 {  101     int days = CalcYearRestDays(ys, ms, ds);  102  103     if(ys != ye) /\*不是同一年的日期\*/  104     {  105         if((ye - ys) >= 2) /\*间隔超过一年，要计算间隔的整年时间\*/  106         {  107             days += CalcYearsDays(ys + 1, ye);  108         }  109         days += CalcYearPassedDays(ye, me, de);  110     }  111     else  112     {  113         days = days - CalcYearRestDays(ye, me, de);  114     }  115  116     return days;  117 }  43 /\*计算一年中过去的天数，包括指定的这一天\*/  44 int CalcYearPassedDays(int year, int month, int day)  45 {  46     int passedDays = 0;  47  48     int i;  49     for(i = 0; i < month - 1; i++)  50     {  51         passedDays += daysOfMonth[i];  52     }  53  54     passedDays += day;  55     if((month > 2) && IsLeapYear(year))  56         passedDays++;  57  58     return passedDays;  59 }  60  61 /\*计算一年中还剩下的天数，不包括指定的这一天\*/  62 int CalcYearRestDays(int year, int month, int day)  63 {  64     int leftDays = daysOfMonth[month - 1] - day;  65  66     int i;  67     for(i = month; i < MONTHES\_FOR\_YEAR; i++)  68     {  69         leftDays += daysOfMonth[i];  70     }  71  72     if((month <= 2) && IsLeapYear(year))  73         leftDays++;  74  75     return leftDays;  76 }  77  78 /\*  79 计算years年1月1日和yeare年1月1日之间的天数，  80 包括years年1月1日，但是不包括yeare年1月1日  81 \*/  82 int CalcYearsDays(int years, int yeare)  83 {  84     int days = 0;  85  86     int i;  87     for(i = years; i < yeare; i++)  88     {  89         if(IsLeapYear(i))  90             days += DAYS\_OF\_LEAP\_YEAR;  91         else  92             days += DAYS\_OF\_NORMAL\_YEAR;  93     }  94  95     return days;  96 } |

        另一种计算两个日期相差天数的方法是利用儒略日（Julian Day，JD）进行计算。首先介绍一下儒略日，儒略日是一种不记年，不记月，只记日的历法，是由法国学者Joseph Justus Scaliger（1540－1609）在1583年提出来的一种以天数为计量单位的流水日历。儒略日和儒略历（Julian Calendar）没有任何关系，命名为儒略日也仅仅他本人为了纪念他的父亲――意大利学者Julius Caesar Scaliger（1484－1558）。简单来讲，儒略日就是指从公元前4713年1月1日UTC 12:00开始所经过的天数，JD0就被指定为公元前4713年1月1日 12:00到公元前4713年1月2日12:00之间的24小时，依次顺推，每一天都被赋予一个唯一的数字。例如从1996年1月1日12:00开始的一天就是儒略日JD2450084。使用儒略日可以把不同历法的年表统一起来，很方便地在各种历法中追溯日期。如果计算两个日期之间的天数，利用儒略日计算也很方便，先计算出两个日期的儒略日数，然后直接相减就可以得到两个日期相隔的天数。

        由公历的日期计算出儒略日数是一个很简单的事情，有多个公式可以计算儒略日，本文选择如下公式计算儒略日：

http://my.csdn.net/uploads/201207/15/1342365145_8797.gif

其中y是年份，m是月份，d是日期，如果m小于或等于2，则m修正为m+12，同时年份修正为y-1。c值由以下方法计算：

http://my.csdn.net/uploads/201207/15/1342365250_2576.gif

下面就是由公历日期计算儒略日的算法实现：

|  |
| --- |
| 119 int CalculateJulianDay(int year, int month, int day)  120 {  121     int B = 0;  122  123     if(month <= 2)  124     {  125         month += 12;  126         year -= 1;  127     }  128     if(IsGregorianDays(year, month, day))  129     {  130         B = year / 100;  131         B = 2 - B + year / 400;  132     }  133  134     double dd = day + 0.5000115740; /\*本日12:00后才是儒略日的开始(过一秒钟)\*/  135     return int(365.25 \* (year + 4716) + 0.01) + int(30.60001 \* (month + 1)) + dd+ B - 1524.5;  136 } |

        儒略日的计算通常精确到秒，得到的JD数也是一个浮点数，本文仅仅是为了计算日期相隔的整数天数，因此都采用整数计算。由于儒略日的周期开始与每天中午12:00，而历法中的天数通常是从0:00开始的，因此儒略日计算上对日期的天数进行了修正。1977年3月27日的儒略日是2443230，2005年5月31日的儒略日是2453522，差值是10292，和前一种方法计算的结果一致。

        我们用两种方法计算出两个日期之间的天数都是10292，现在用10292除以7得到余数是2，也就是说2005年5月31日与1977年3月27日星期数差两天，所以2005年5月31日就是是星期二。

        上述计算星期的方法虽然步骤简单，但是每次都要计算两个日期的时间差，不是非常方便。如果能够有一个公式可以直接根据日期计算出对应的星期岂不是更好？幸运的是，这样的公式是存在的，下篇将继续介绍公式法直接计算某一天星期数的算法。

小知识1：公历的闰年

中国公历（也就是格里历）的置闰规则是四年一闰，百年不闰，四百年再闰，为什么会有这么奇怪的置闰规则呢？这实际上与天体运行周期与人类定义的历法周期之间的误差有关。地球绕太阳运转的周期是365.2422天，即一个回归年（Tropical Year），而公历的一年是365天，这样一年就比回归年短了0.2422日，四年积累下来就多出0.9688天（约1天），于是设置一个闰年，这一年多一天。这样一来，四个公历年又比四个回归年多了0.0312天，平均每年多0.0078天，这样经过四百年就会多出3.12天，也就是说每四百年要减少3个闰年才行，于是就设置了百年不闰，四百年再闰的置闰规则。

实际上公历的置闰还有一条规则，就是对于数值很大的年份，如果能整除3200，同时能整除172800则是闰年。这是因为前面即使四百年一闰，仍然多了0.12天，平均就是每年多0.0003天，于是每3200年就又多出0.96天，也就是说每3200年还要减少一个闰年，于是能被3200整除的年就不是闰年了。然而误差并没有终结，每3200年减少一个闰年（减少一天）实际上多减了0.04天，这个误差还要继续累计计算，这已经超出了本文的范围，有兴趣的读者可以自己计算。

[算法系列之十七：日历生成算法-中国公历（格里历）（下）](http://blog.csdn.net/orbit/article/details/7825004)

【接上篇】

        上述计算星期的方法虽然步骤简单，但是每次都要计算两个日期的时间差，不是非常方便。如果能够有一个公式可以直接根据日期计算出对应的星期岂不是更好？幸运的是，这样的公式是存在的。此类公式的推导原理仍然是通过两个日期的时间差来计算星期，只是通过选择一个特殊的日期来简化公式的推导。这个所谓的特殊日期指的是某一年的12月31日这天刚好是星期日这种情况。选择这样的日子有两个好处，一个是计算上可以省去计算标准日期这一年的剩余天数，另一个是计算出来的日期差余数是几就是星期几，不需要再计算星期的差值。人们知道公元元年的1月1日是星期一，那么公元前1年的12月31日就是星期日，用这一天作为标准日期，就可以只计算整数年的时间和日期所在的年积累的天数，这个星期公式就是：

w = (L \* 366 + N \* 365 + D) % 7                             (公式 2)

公式中的L是从公元元年到y年m月d日所在的年之间的闰年次数，N是平常年次数，D是y年内的积累天数。将整年数y - 1 = L + N带入上式，可得：

w = ( (y - 1) \* 365 + L + D) % 7                              (公式 3)

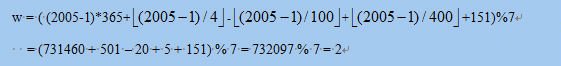
根据闰年规律，从公元元年到y年之间的闰年次数是可以计算出来的，即：

http://my.csdn.net/uploads/201208/02/1343919135_4487.gif

将L带入公式2，得到星期w的最终计算公式：

http://my.csdn.net/uploads/201208/02/1343919189_4239.gif

还以2005年5月31日为例，利用公式5计算w的值为：



得到2005年5月31日是星期二，和前面的计算方法得到的结果一致。根据上述分析，可得写出使用公式5计算星期的算法实现：

|  |
| --- |
| 146 int TotalWeek(int year, int month, int day)  147 {  148     int d = CalcYearPassedDays(year, month, day);  149     int y = year - 1;  150     int w = y \* DAYS\_OF\_NORMAL\_YEAR + y / 4 - y / 100 + y / 400 + d;  151  152     return w % 7;  153 } |

        公式5的问题在于计算量大，不利于口算星期结果。于是人们就在公式5的基础上继续推导更简单的公式。德国数学家克里斯蒂安·蔡勒（Christian Zeller, 1822- 1899）在1886年推导出了著名的为蔡勒（Zeller）公式：

http://my.csdn.net/uploads/201208/02/1343919369_9142.gif

对计算出的w值除以7，得到的余数就是星期几，如果余数是0，则为星期日。蔡勒公式中各符号的含义如下：

w ：星期；

c ：世纪数 – 1的值，如21世纪，则 = 20；

m ：月数，的取值是大于等于3，小于等于14。在蔡勒公式中，某年的1月和2月看作上一年的13月和14月，比如2001年2月1日要当成2000年的14月1日计算；

y ：年份，取公元纪念的后两位，如1998年， = 98，2001年， = 1；

d ：某月内的日数

为了方便口算，人们通常将公式6中的http://my.csdn.net/uploads/201208/02/1343919484_9342.gif一项改成http://my.csdn.net/uploads/201208/02/1343919519_2640.gif

。目前人们普遍认为蔡勒公式是计算某一天是星期几的最好的公式。但是蔡勒公式有时候可能计算出的结果是负数，需要对结果+7进行修正。比如2006年7月1日，用蔡勒公式计算出的结果是 -1，实际上这天是星期六。根据前面分析的结果整理出的蔡勒公式算法实现如下：

|  |
| --- |
| 155 int ZellerWeek(int year, int month, int day)  156 {  157     int m = month;  158     int d = day;  159  160     if(month <= 2) /\*对小于2的月份进行修正\*/  161     {  162         year--;  163         m = month + 12;  164     }  165  166     int y = year % 100;  167     int c = year / 100;  168  169     int w = (y + y / 4 + c / 4 - 2 \* c + (13 \* (m + 1) / 5) + d - 1) % 7;  170     if(w < 0) /\*修正计算结果是负数的情况\*/  171         w += 7;  172  173     return w;  174 } |

        蔡勒公式（公式6）和前面提到的公式5都只适用于格里历法。罗马教皇在1582年修改历法，将10月5日指定为10月15日，从而正式废止儒略历法，开始启用格里历法。因此，上述求星期几的公式只适用于1582年10月15日之后的日期，对于1582年将10月4日之前的日期，蔡勒也推导出了适用与儒略历法的星期计算公式：

http://my.csdn.net/uploads/201208/02/1343919635_3154.gif

公式7适用于对1582年10月4日之前的日期计算星期，1582年10月5日与1582年10月15日之间的日期是不存在的，因为它们都是同一天。

        格里历历法简单，除二月外每月天数固定，二月则根据是否是闰年确定是28天还是29天，每天的星期数可以通过蔡勒公式（公式6）计算，有了这些信息，就可以按照一定的排版格式将某一年的日历打印出来。排版打印的算法非常简单，就是按照顺序打印12个月的月历，因此，打印月历的函数就是输出算法的重点。代码没什么特别之处，就是用一些小技巧确定每个月的第一天的开始位置，打印月历的核心代码如下：

|  |
| --- |
| 229 void PrintMonthCalendar(int year, int month)  230 {  231     int days = GetDaysOfMonth(year, month); /\*确定这个月的天数\*/  232     if(days <= 0)  233         return;  234  235     PrintMonthBanner(nameOfMonth[month - 1]);  236     PrintWeekBanner();  237     int firstDayWeek = ZellerWeek(year, month, 1);  238     InsertRowSpace(firstDayWeek);  239     int week = firstDayWeek;  240     int i = 1;  241     while(i <= days)  242     {  243         printf("%-10d", i);  244         if(week == 6) /\*到一周结束，切换到下一行输出\*/  245         {  246             SetNextRowStart();  247         }  248         i++;  249         week = (week + 1) % 7;  250     }  251 } |

GetDaysOfMonth()函数其实就是从daysOfMonth表中查一下每月的天数，如果是闰年，则对二月的天数修正（＋1），daysOfMonth表定义如下：

int daysOfMonth[MONTHES\_FOR\_YEAR] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

计算星期不必对每一天都计算一次，只要对每个月的第一天计算一次就可以了，以后的日期可以用 week = (week + 1) % 7 直接推算出星期几。下面就是我们的算法打印输出的效果：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                              Calendar of 2012

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

----------January----------

Sunday    Monday    Tuesday   Wednesday Thursday  Friday    Saturday

1         2         3         4         5         6         7

8         9         10        11        12        13        14

15        16        17        18        19        20        21

22        23        24        25        26        27        28

29        30        31

----------February----------

Sunday    Monday    Tuesday   Wednesday Thursday  Friday    Saturday

                              1         2         3         4

5         6         7         8         9         10        11

12        13        14        15        16        17        18

19        20        21        22        23        24        25

26        27        28        29

----------March----------

Sunday    Monday    Tuesday   Wednesday Thursday  Friday    Saturday

                                        1         2         3

4         5         6         7         8         9         10

11        12        13        14        15        16        17

18        19        20        21        22        23        24

25        26        27        28        29        30        31

……

**小知识2：**儒略历和格里历

在公元1582年10月15日之前，人们使用的历法是源自古罗马的儒略历，儒略历的置闰规则就是四年一闰，但是没有计算每年多出来的0.0078天，这样从公元前46年到公元1582年一共累积多出了10天，为此，当时的教皇格里十三世将1582年10月5日人为指定为10月15日，并开始启用新的置闰规则，这就是后来沿用至今的格里历。

**小知识3：**约化儒略日

由于儒略日数字位数太多，国际天文联合会于1973年8月决定对其修正，采用约化儒略日（MJD）进行天文计算，定义MJD = JD – 2400000.5，MJD相应的起始点是1858年11月17日 0:00。

**小知识4：**1752年9月到底是怎么回事儿

如果你用的操作系统是unix或linux，在控制台输入以下命令：

#cal 9 1752

你会看到这样一个奇怪的月历输出：

September 1752

Su Mo Tu We Th Fr Sa

       1  2 14 15 16

17 18 19 20 21 22 23

24 25 26 27 28 29 30

1752年的9月缺了11天，到底怎么回事儿？这其实还是因为从儒略历到格里历的转换造成的。1582年10月5日，罗马教皇格里十三世宣布启用更为精确的格里历，但是整个欧洲大陆并不是所有国家都立即采用格里历，比如大英帝国就是直到1752年9月议会才批准采用格里历，所以大英帝国及其所有殖民地的历法一直到1752年9月才发生跳变，“跟上”了格里历。德国和荷兰到了1698年才采用格里历，而俄罗斯则直到1918年革命才采用格里历。Linux的cal指令起源与最初AT&T的UNIX，当然采用的是美国历法，但是美国历史太短，再往前就只能采用英国历法，所以cal指令的结果就成了这样。对于采用格里历的国家来说，只要知道1582年10月发生了日期跳变就行了，可以不用关心1752年9月到底是怎么回事儿。但是对于研究历史和考古的人来说，就必需要了解这个历史，搞清楚每个欧洲国家改用格里历的年份，否则就可能在一些问题上出错。在欧洲研究历史，你会发现很多事件都是有多个时间版本的，比如大科学家牛顿的生日就有两个时间版本，一个是按照儒略历历法的1642年12月25日，另一个是格里历历法的1643年1月4日，对于英国人来说，1752年之前都是按照儒略历计算的，所以英国的史书可能会记载牛顿出生在圣诞节，这也没什么可奇怪的。