

电子信息与通信学院

实 验 报 告

|  |  |
| --- | --- |
| 实验名称 | 课程综合练习 |
| 课程名称 | 计算机基础  与程序设计(C) |

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 王威 | 学号 | U202416551 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2024.12.19 | 地点 | 华中科技大学 |

|  |  |  |  |
| --- | --- | --- | --- |
| 成绩 |  | 教师 | 刘威 |

# 实验目的

完成日历系列代码。

# 实验环境

操作系统：Windows 10

编程工具：CodeBlocks 16.01

# 实验一

## 实验任务

运用所学到的知识，编写一个计算星期几的程序，计算2023年某月某日是星期

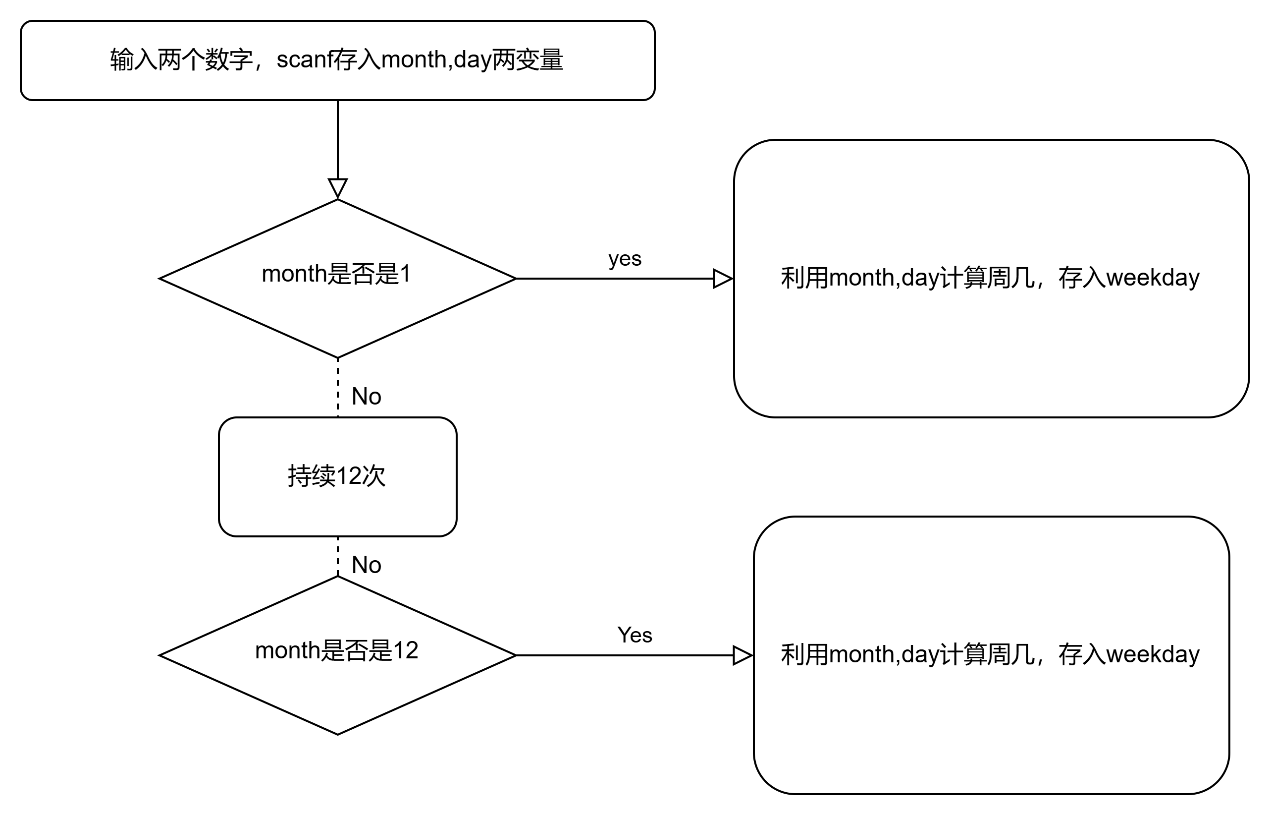
几（已知2023年1月1日是星期日）

• 设置一个变量计算输入日是全年第几天，通过算术取模运算计算输入日是星期几

• 约定每个星期从周一开始，如果是周一则打印1，周二则打印2，……，周日打印7

• 提示：需要提前学习if 语句，不需要用数组

## 实验步骤



关键代码：

int month, day, integer1;

puts("请输入月与日的数字，中间空一格");

scanf ("%d %d",& month,& day );

if( month == 12 )

integer1 = ( day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 ) % 7;

if ( integer1 == 0 )

integer1 = integer1 + 7;

printf ( "%d月%d号是周%d", month, day, integer1 );

## 代码测试

### 测试点 “是否为x月”的测试结果

预期：如果是，就可以根据，month，与day计算出星期几并输出

比如输入2 6，会得到周2

-->



测试无问题

## 实验结论

代码达到功能目标

## 实验总结

由于这是我很早之前的代码，所以只用if的话就显得很臃肿，不过问题很简单，所以测试方面没有什么问题。

# 实验二

## 实验任务

运用所学到的知识，编写打印月历的程序，打印2021年1到12月的某个月的月

历（已知2021年1月1日周五）

• 约定每个星期从周一开始

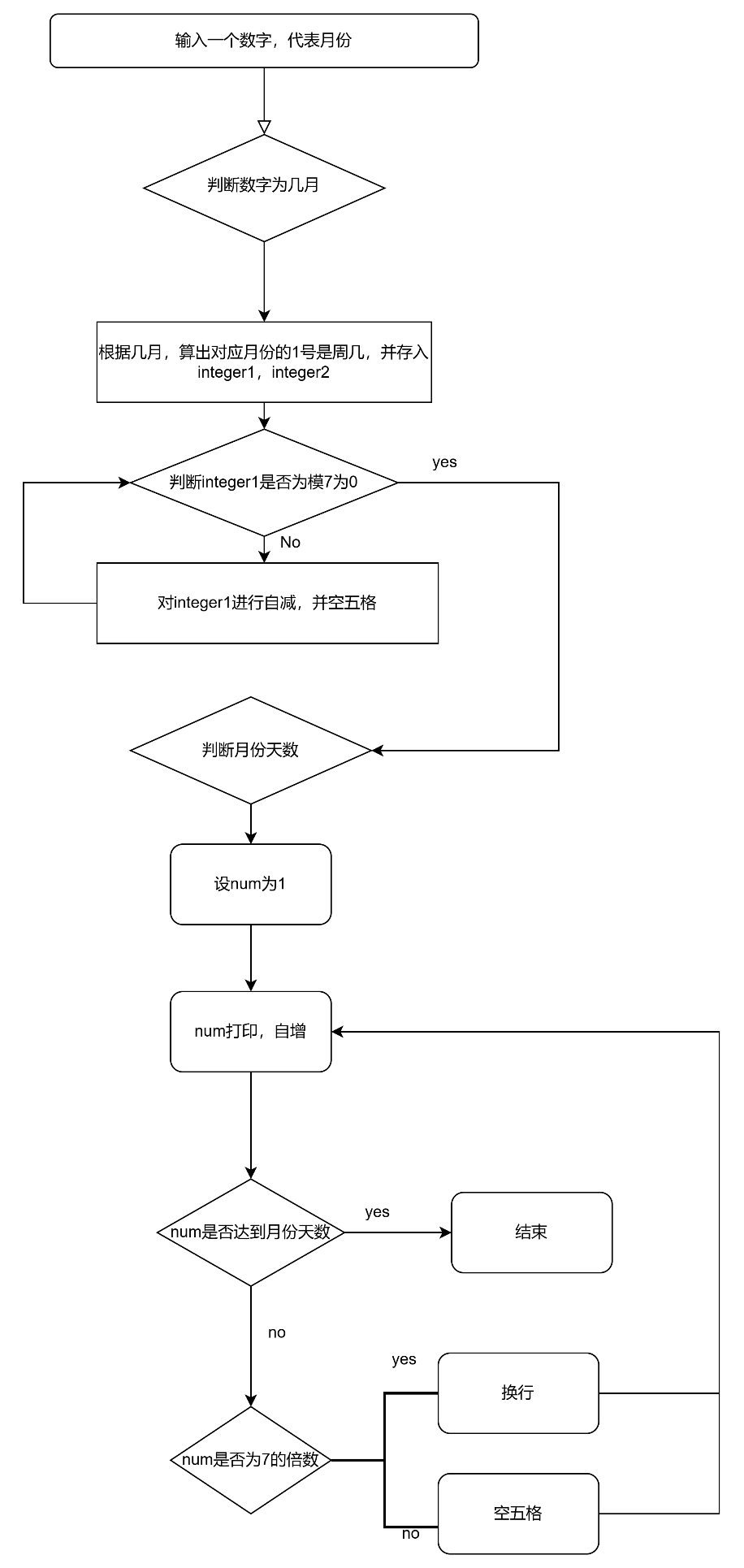
• 约定月历的每列的宽度为10个字符，可以在 printf 语句中用%10s打印空格、用

%10d打印数字来定位

• 【提示】设置一个变量存储每个月的总天数；设置一个变量存储第一天是星期几（利用

前一个实验计算）；利用循环控制语句打印该月的每一天，如果碰到周日要换行

## 实验步骤



关键代码：

printf ("%10s%10s%10s%10s%10s%10s%10s\n",

"Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");

//先判断月份

if ( month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12 )

{

while ( ( integer1 - 1 ) != 0 )

{

printf(" ");

integer1 = integer1 - 1;

}

while ( num <= 31 )

{

if ( (num + integer2 - 1 ) % 7 != 0 )

{

printf( "%10d",num );

num = num + 1;

}

else

{

printf ( "%10d\n",num );

num = num + 1;

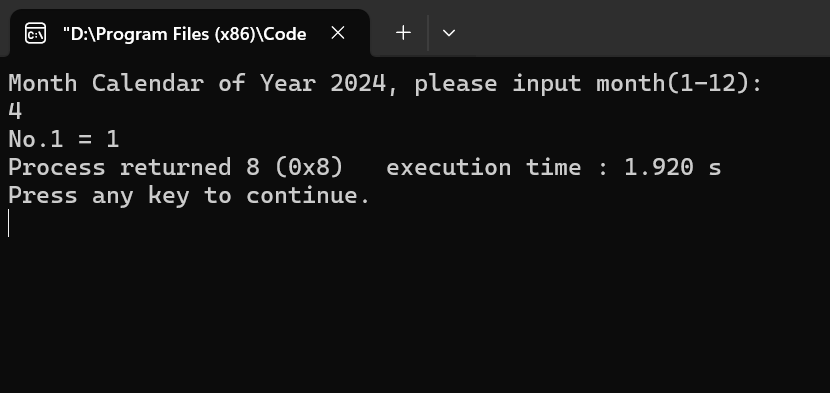
}

}

}

## 代码测试

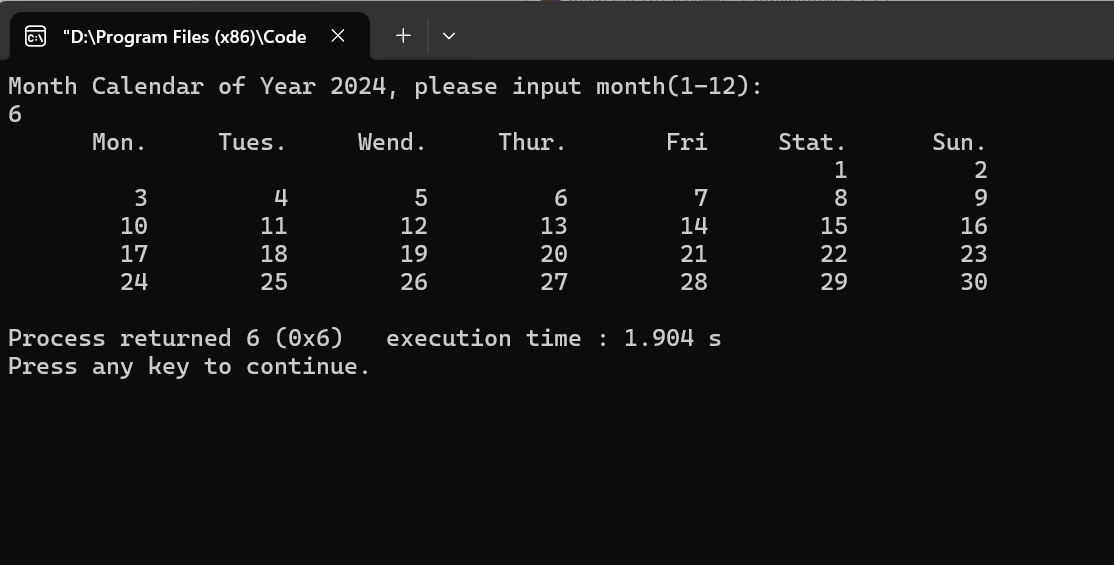
### 测试点 “1号为周几”的测试结果



预期：输入月份，可以输出1号为周几

结论：没问题

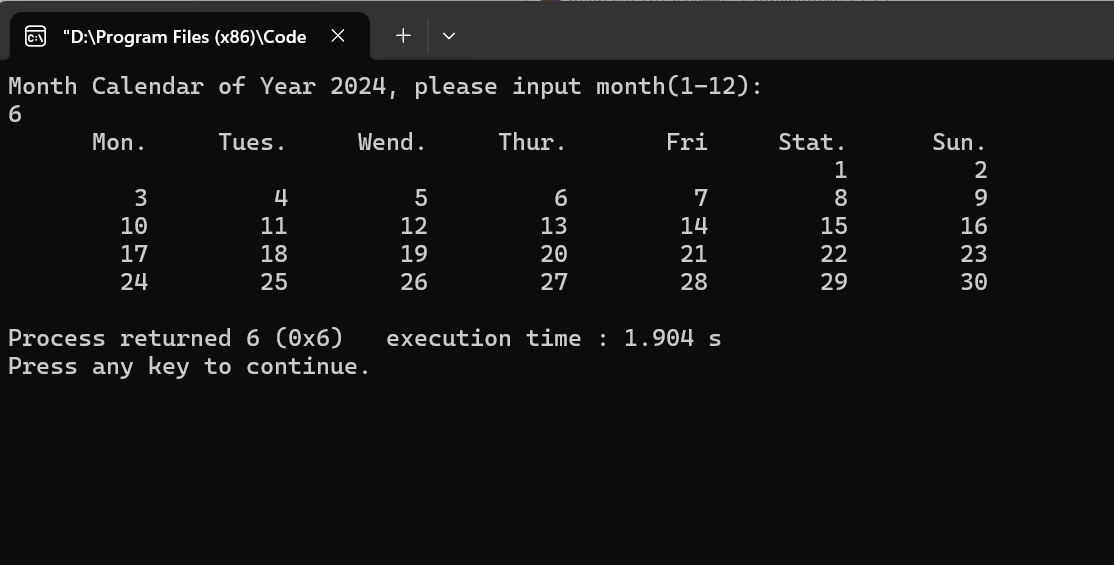
### 测试点 “是否能integer1自减空相应格”的测试结果



预期结果：integer1=6，就空五次

结论：测试无问题

### 测试点 “是否num自增，且为7倍数就换行，且达到当月月份停止”的测试结果



预期结果：比如月份为六，达到三十就停止

结果：为7倍数就换行，且达到30停止

## 实验结论

代码达到功能目标

## 实验总结

由于这是我很早之前的代码，当时也是花了好久一点点磨出来的，当时写完没有问题还去找老师炫耀，哈哈。

# 实验三

## 实验任务

运用所学到的知识，编写一个打印周历的程序，打印2021年某一周的周历

（2021年1月1日周四）

– 约定每个星期从周一开始

– 约定周历的每列的宽度为10个字符，可以在 printf 语句中用%5s打印空格、

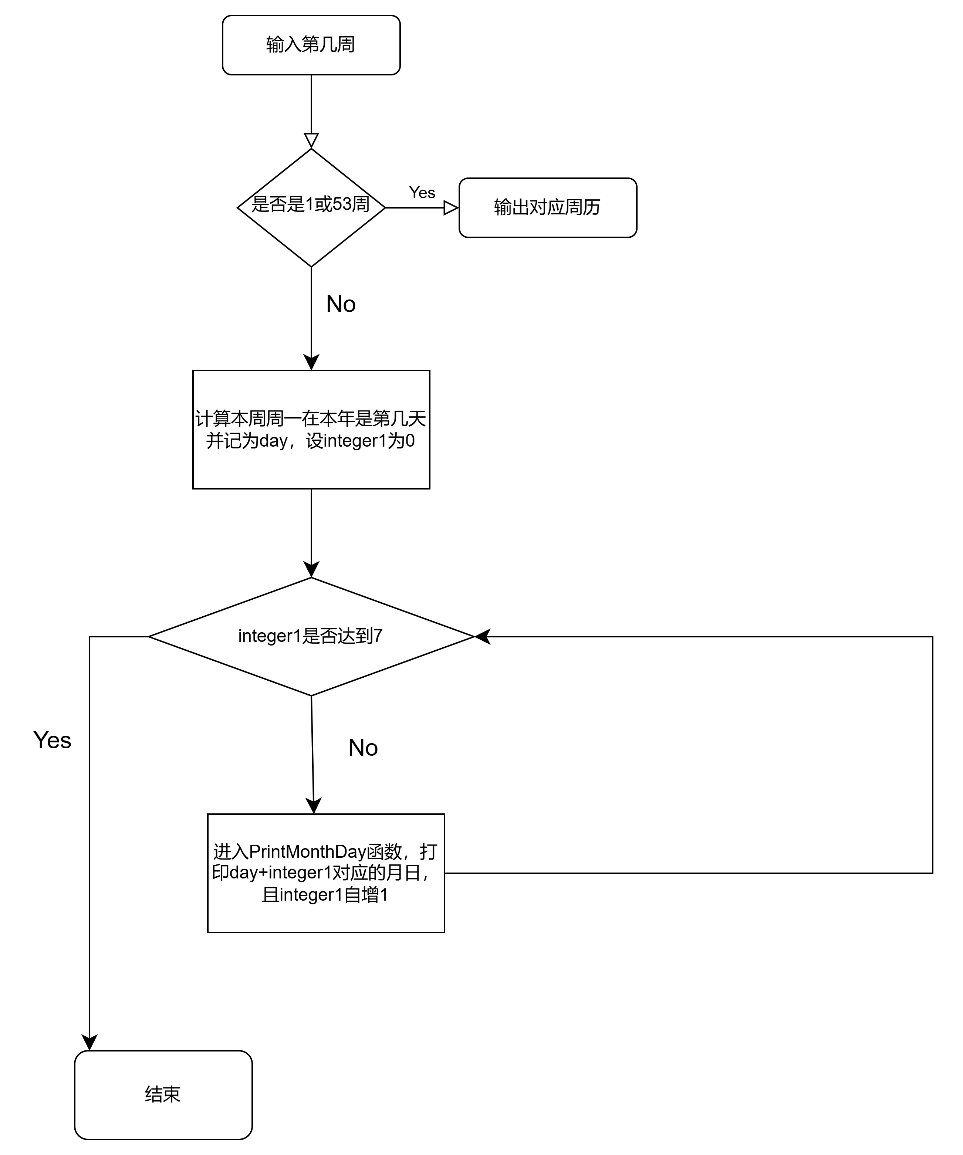
用%2d打印数字来定位

– 第1周和第53周中，仅打印2021年的日期

– 【提示】先求出该周的起始日、结束日是全年的第几天，以全年第几天为循环

变量打印周历，循环体中判断所打印日期是几月几号

## 实验步骤



关键代码：

if ( week > 1 && week < 53 )

{

day = week \* 7 - 3;

int integer1 = 0;//用来遍历一周

printf (" %02d:", week );

for ( ; integer1 < 7; integer1 ++ )

{

PrintMonthDay ( day + integer1 );

}

}

else if ( week == 1 )

{

printf ("%10s%10s%10s%10s%10s%10s%10s%10s\n",

"01:"," "," "," ","","01.01","01.02","01.03");

}

else if ( week == 53 )

{

printf ("%10s%10s%10s%10s%10s%10s%10s%10s\n",

"53:","12.27","12.28","12.29","12.30","12.31", " "," ");

}

}

void PrintMonthDay ( int data )

{

int month;

int day;

if ( data <= 31 )

{

month = 1;

day = data;

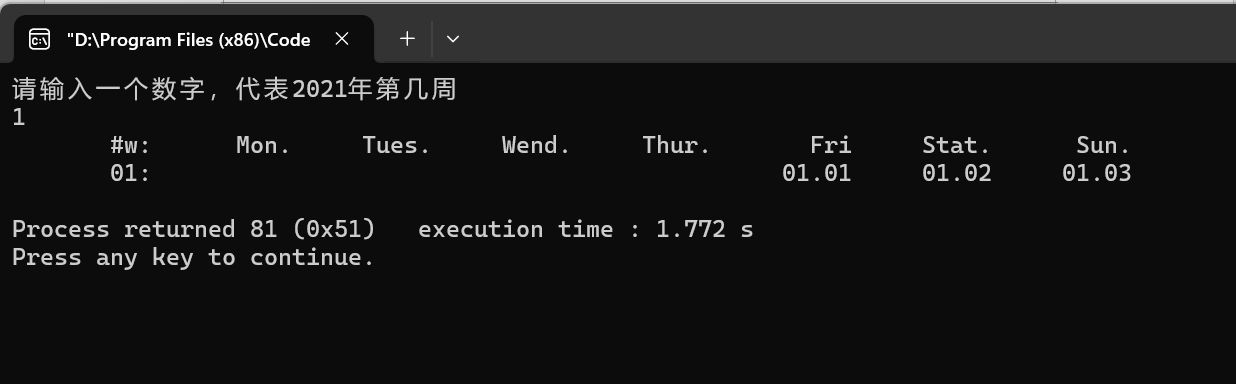
printf (" %02d.%02d", month, day );

return;

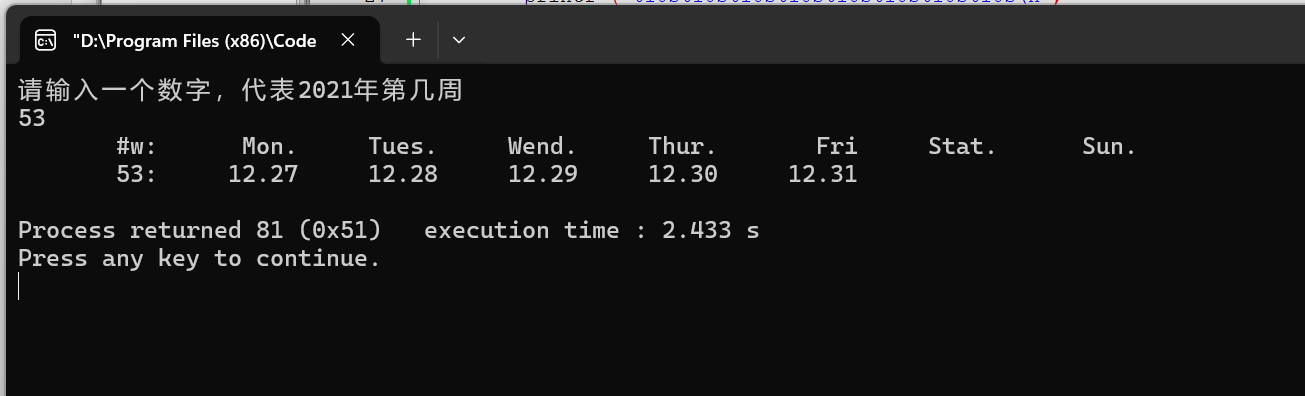
}

## 代码测试

### 测试点 “是否为第1周或第53周”的测试结果

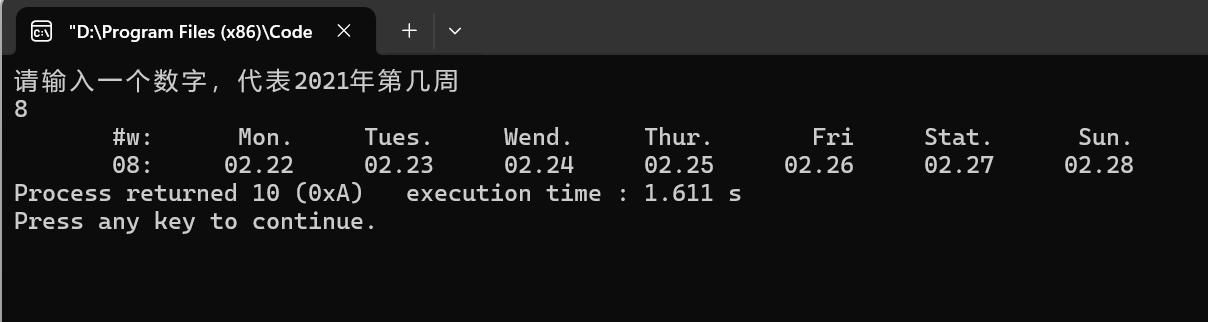


-->预期：输入1或53，可以空相应格



结论：没问题

### 测试点 “是否能在打印一周后停下”的测试结果

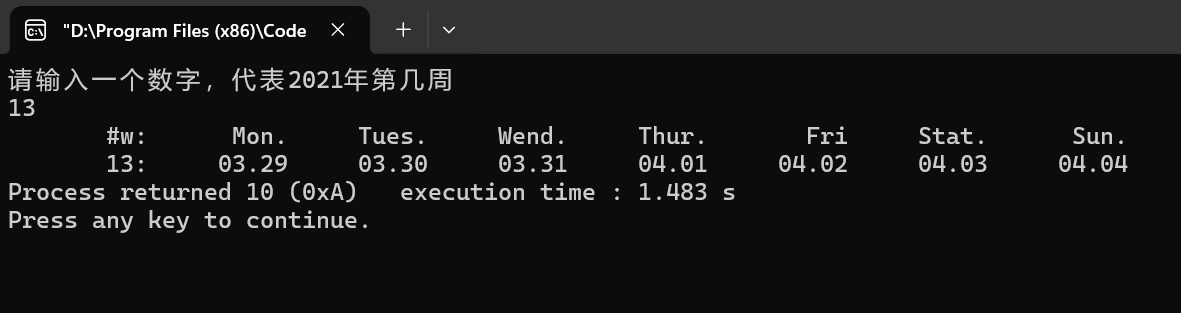


预期结果：在普通周输出7个停下

结论：测试无问题

### 测试点 “在达到月份最大天数后换下一个月份”的测试结果

-->



预期结果：比如月份为3，达到三十一就换下一个月

结果：为7倍数就换行，且达到30停止

## 实验结论

代码达到功能目标

## 实验总结

对2021年周历完成了较好的实现，但是可移植性不高，有待改进

# 实验四

## 实验任务

运用所学到的知识，编写一个打印华中科技大学校历的程序，打印2020年春季学期的

校历

– 约定1：春季学期从2月份的某一周开始，用户输入2月份的某日作为参考日：如果该日

就是周一，即从当周开始；如果该日不是周一，即从下周开始。

– 约定2：春季学期到7月份的第一个周末结束。

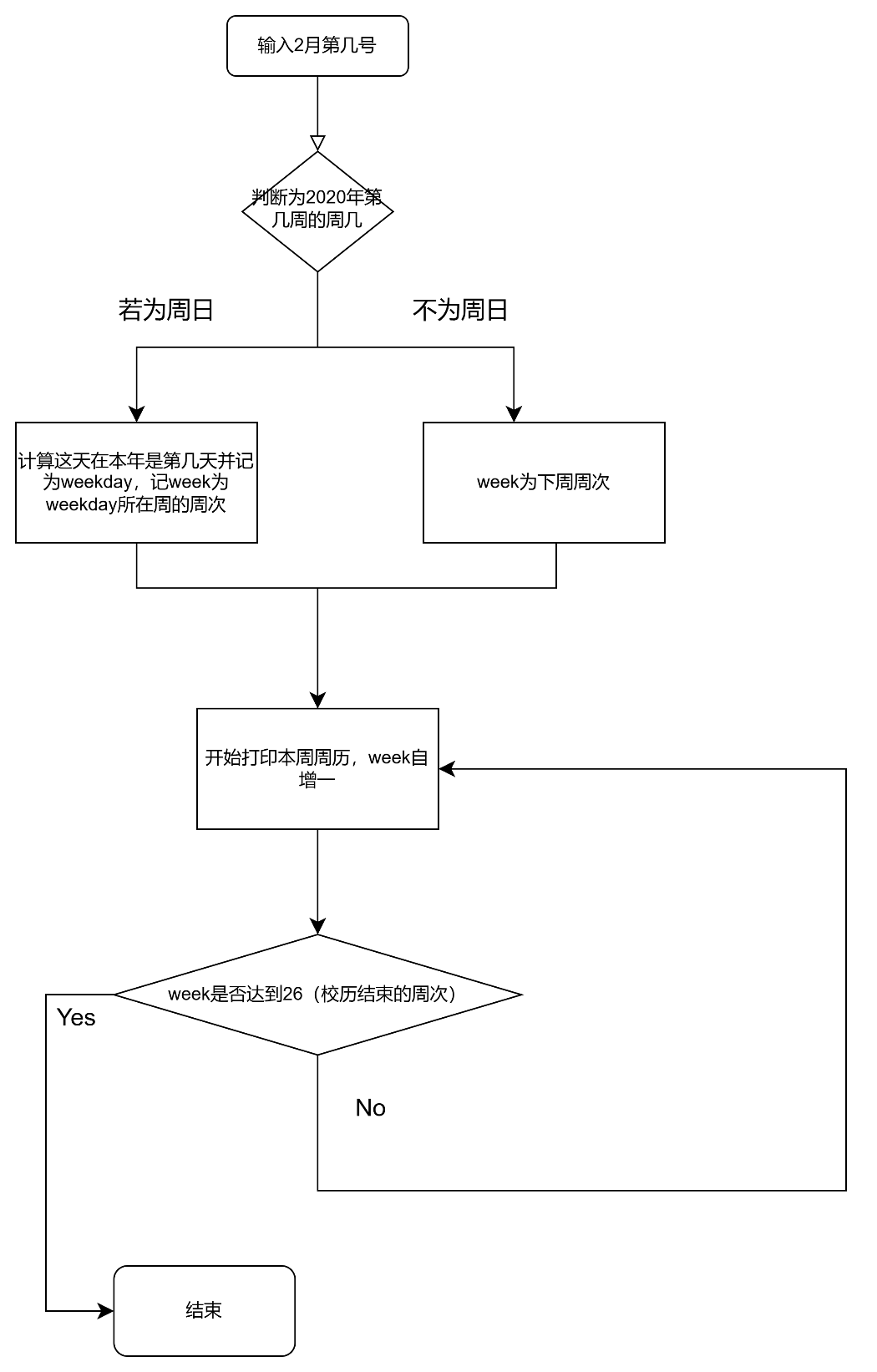
– 约定校历的每列的宽度为10个字符，可以在 printf 语句中用%5s打印空格、用

%02d打印月或者日的数字

– 【提示】编程思路类似周历，以全年第几天为循环变量打印，循环体中判断所打印日

期是几月几号；结合周几的信息，打印周一前的周次，打印周日后的换行

## 实验步骤



关键代码：

for ( ; week <= 26; week ++ )

{

day = week \* 7 - 1;

int integer1 = 0;//用来遍历一周

printf (" %02d:", integerweek );

integerweek ++;

for ( ; integer1 < 7; integer1 ++ )

{

PrintMonthDay ( day + integer1 );

}

printf ("\n");

}

}

void PrintMonthDay ( int data )

{

int month;

int day;

if ( data <= 31 + 29 )

{

month = 2;

day = data - 31;

printf (" %02d.%02d", month, day );

return;

}

if ( data <= 31 + 29 + 31 )

{

month = 3;

day = data - 31 - 29;

printf (" %02d.%02d", month, day );

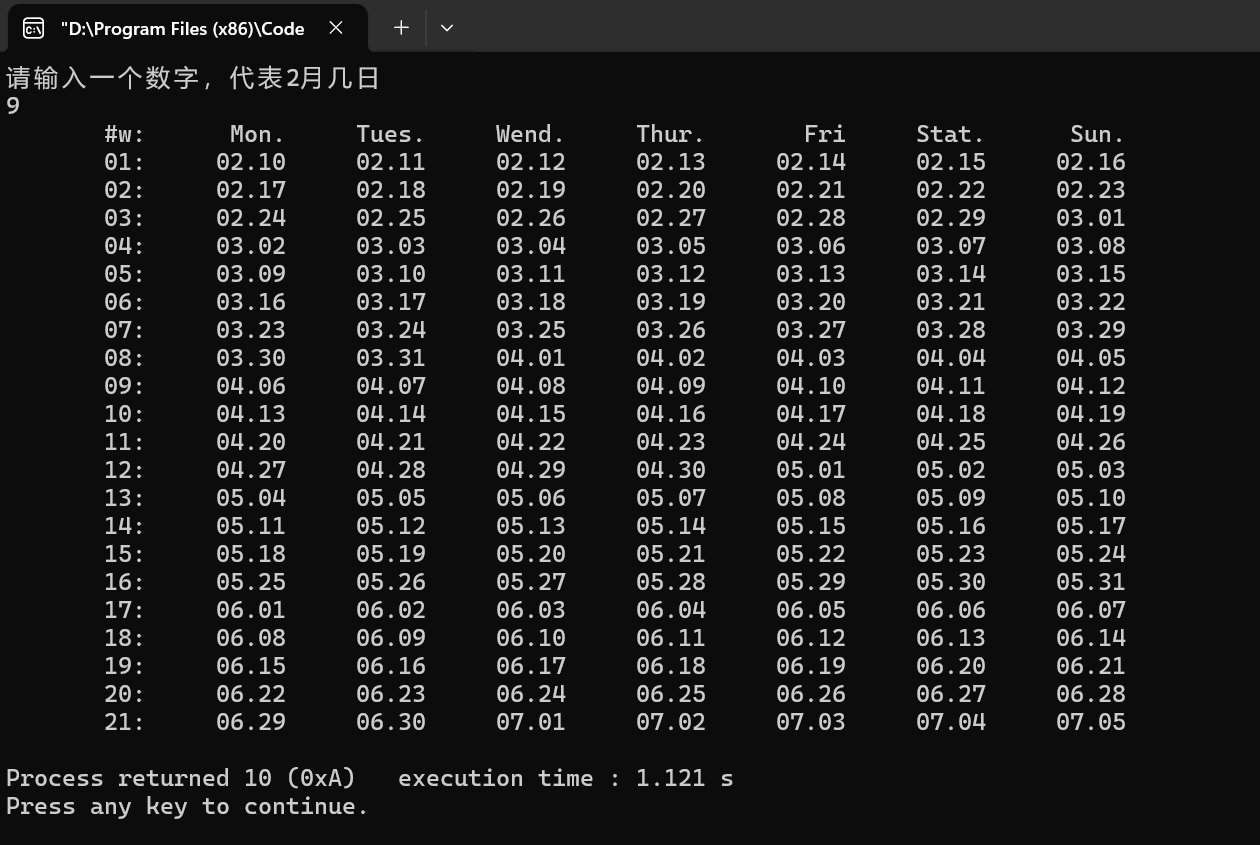
return;

}

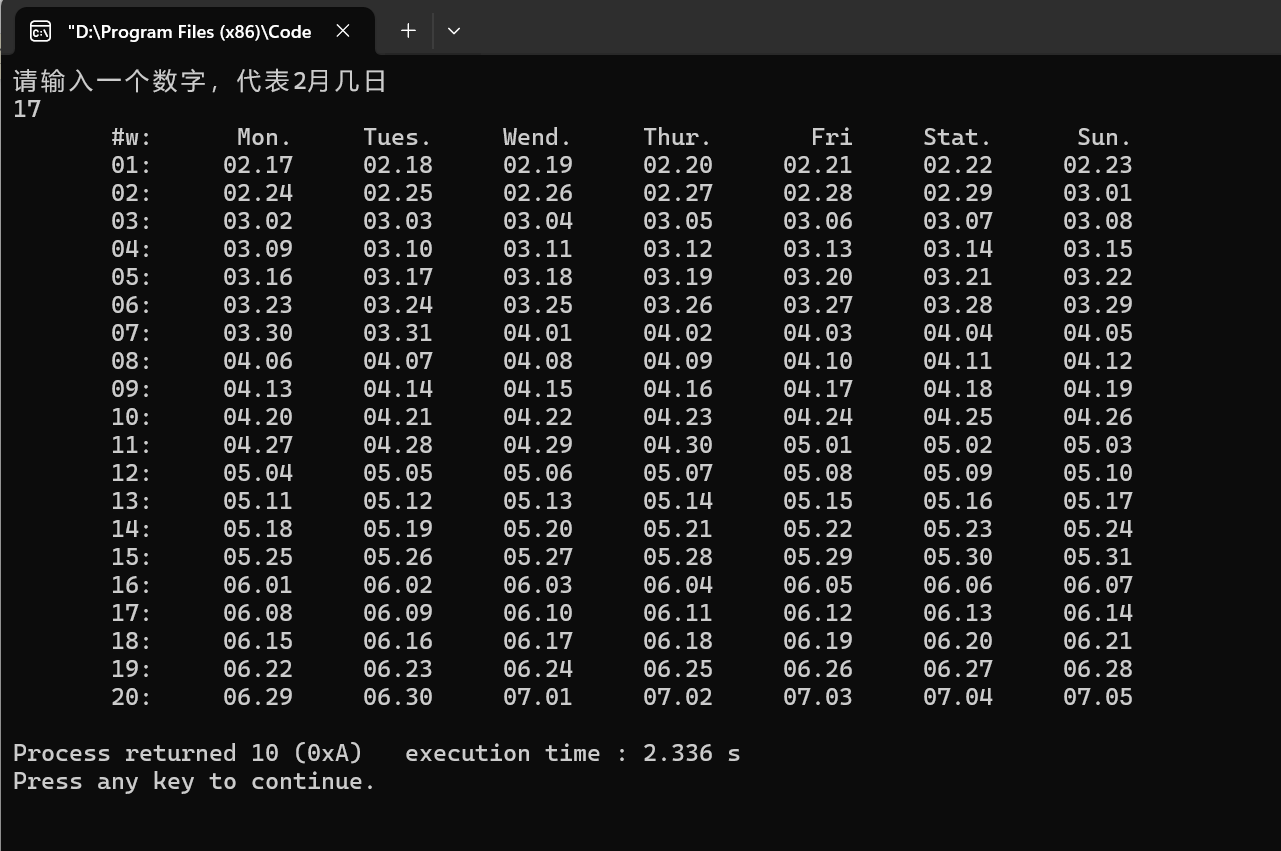
## 代码测试

### 测试点 “能否按照规定选择起始周次”的测试结果

-->



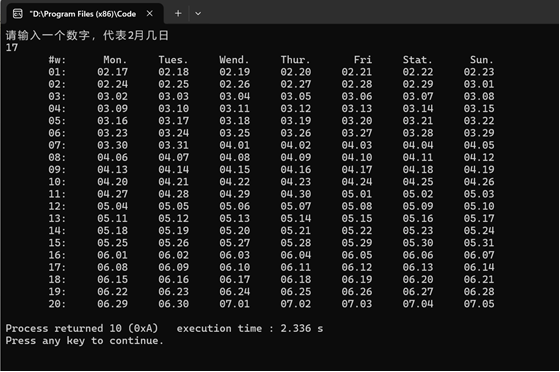
-->



预期：输入周一的日期，从当日开始，其他日期从下周开始

结论：没问题

### 测试点 “能否七天一换行，能否到月底就换月”的测试结果

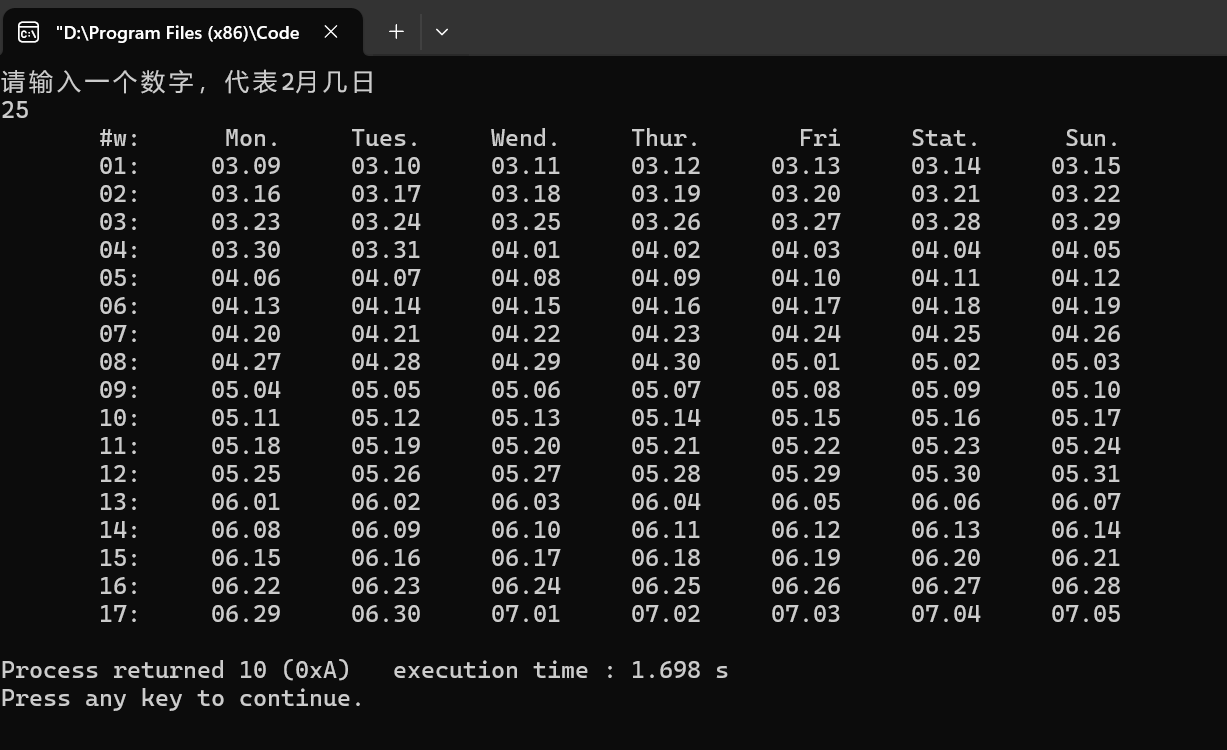


预期结果：七天一换行，到月底就换月

结论：测试无问题

### 测试点 “能否在7.5停止”的测试结果

-->



预期结果：校历到7.5就停止

结果：测试无问题

## 实验结论

代码达到功能目标

## 实验总结

这个问题跟实验三周历其实很像，两个一联系，校历还是很好解决的

# 实验五

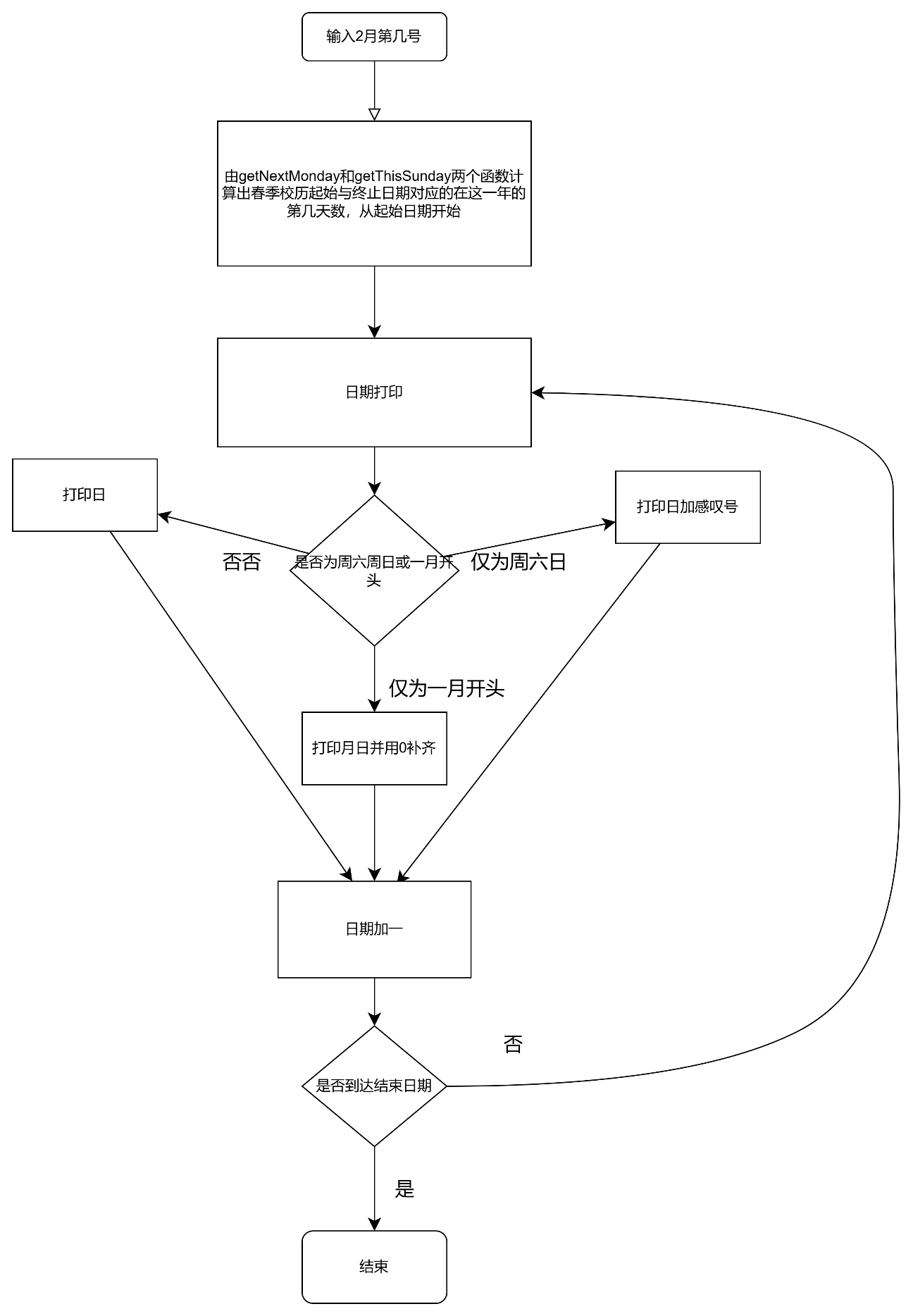
## 实验任务

用函数改写打印华中科技大学校历的程序

打印2021年春季学期的校历

– 要求开发下列函数：

## 实验步骤



关键代码：

int getNextMonday( int daySeqOfYear)

{

if ( getDaySeqOfWeek( daySeqOfYear ) == 1 )

{

return daySeqOfYear;

}

else

{

while ( getDaySeqOfWeek( daySeqOfYear ) != 1 )

{

daySeqOfYear ++;

}

return daySeqOfYear;

}

}

int getThisSunday( int daySeqOfYear)

{

if ( getDaySeqOfWeek( daySeqOfYear ) == 7 )

{

return daySeqOfYear;

}

else

{

while ( getDaySeqOfWeek( daySeqOfYear ) != 7 )

{

daySeqOfYear ++;

}

return daySeqOfYear;

}

}

int getMonth( int daySeqOfYear)

{

int data = daySeqOfYear;

int month;

int day;

if ( data <= 31 )

{

month = 1;

day = data;

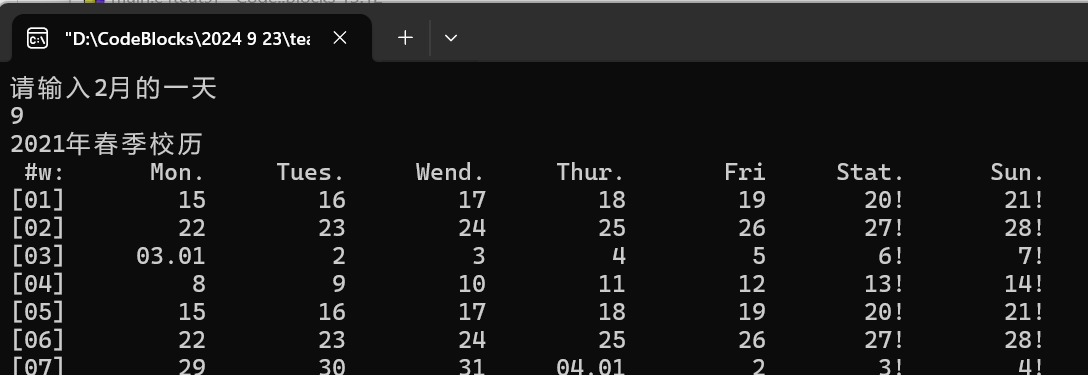
return month;

}

## 代码测试

### 测试点 “是否可以输入起始日期得到合适的开始”的测试结果

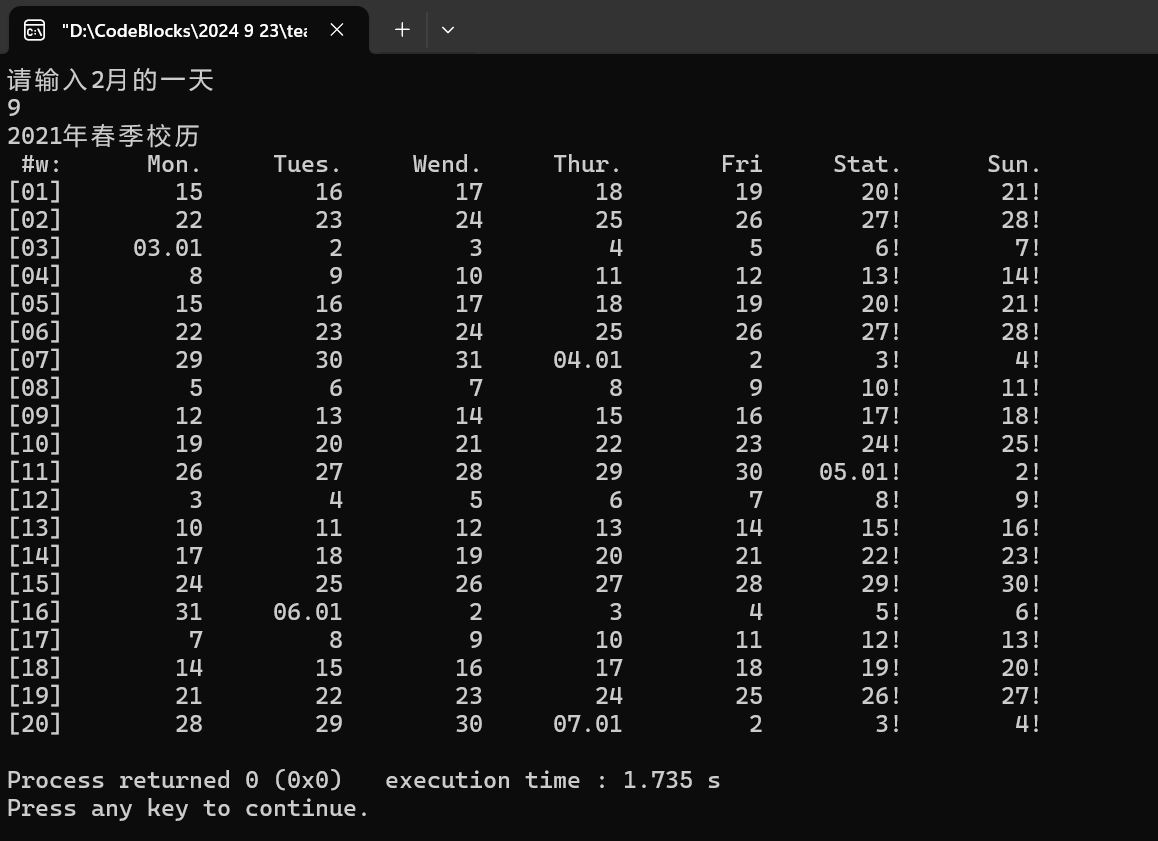
-



预期：输入日期，得到本周或下周周一开始的校历（若输入日期为周一则从本周一开始，不是则下周一）

结论：没问题

### 测试点 “是否能在周六周日后面加感叹号，每月1号改成0x.0x的形式”的测试结果

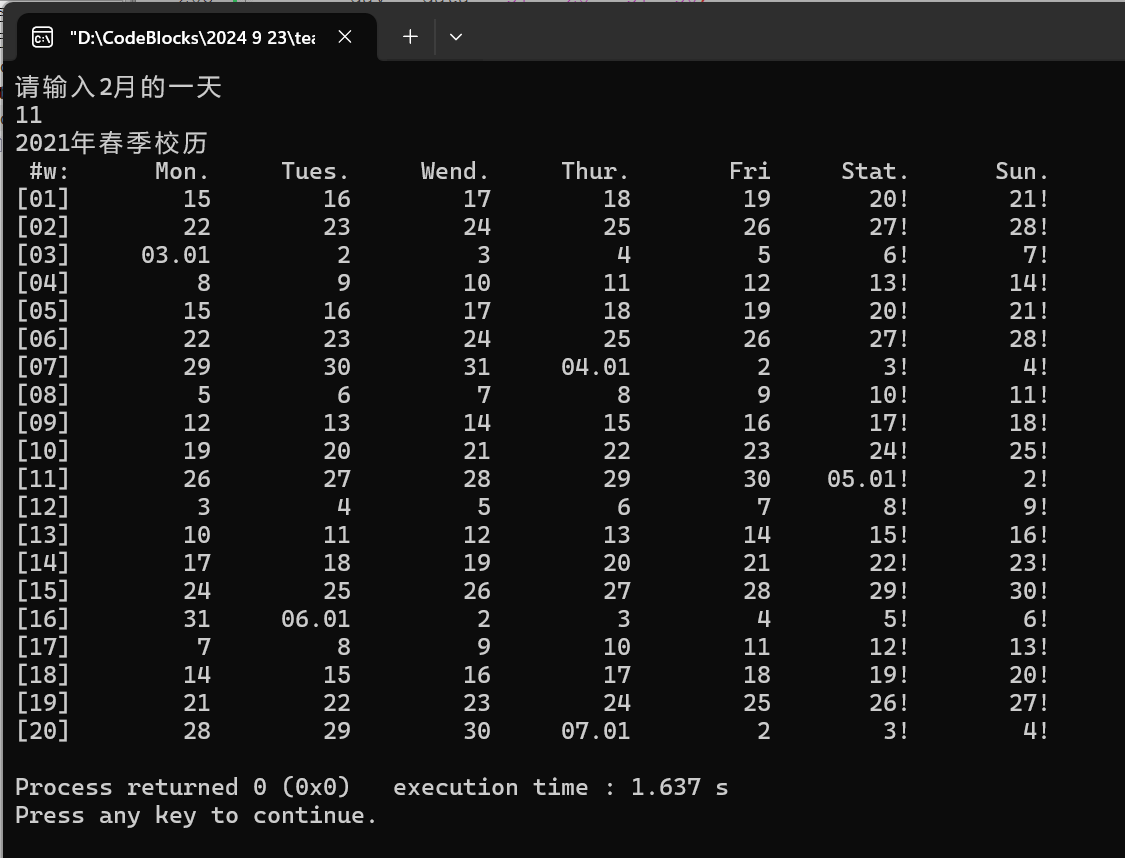


预期结果： 周六周日后面加感叹号，每月1号改成0x.0x的形式

结论：测试无问题

### 测试点 “是否能在7月第一个周末停止”的测试结果

-->-->



预期结果：比如2021年在7月4日停止

结果：测试无问题

## 实验结论

代码达到功能目标

## 实验总结

这题题目给的函数特别多，需要依次缕清哪个函数是干什么的，哪个函数是为哪个函数服务的，不过之前题目打下了基础，这个也不是太难，花了两个小时完成了，跑代码时特别无语，因为本身这个题代码很大，我写出来觉得没问题但是它一直报错，我再细细找就要很长时间了，更离奇的是，报错的行是我根本没用的行，然后我想可能是折叠了或者怎么，然后我就换了一个存储的地方，这次果然能跑了，但是也有问题，然后我就从起始终止入手，发现起始日期有问题，一改，好了。觉得收获很大。

# 实验六

## 实验任务

用函数改写打印华中科技大学校历的程序，打

印2021年秋季学期的校历

– 秋季学期从9月1日所在周的周一开始，到第二

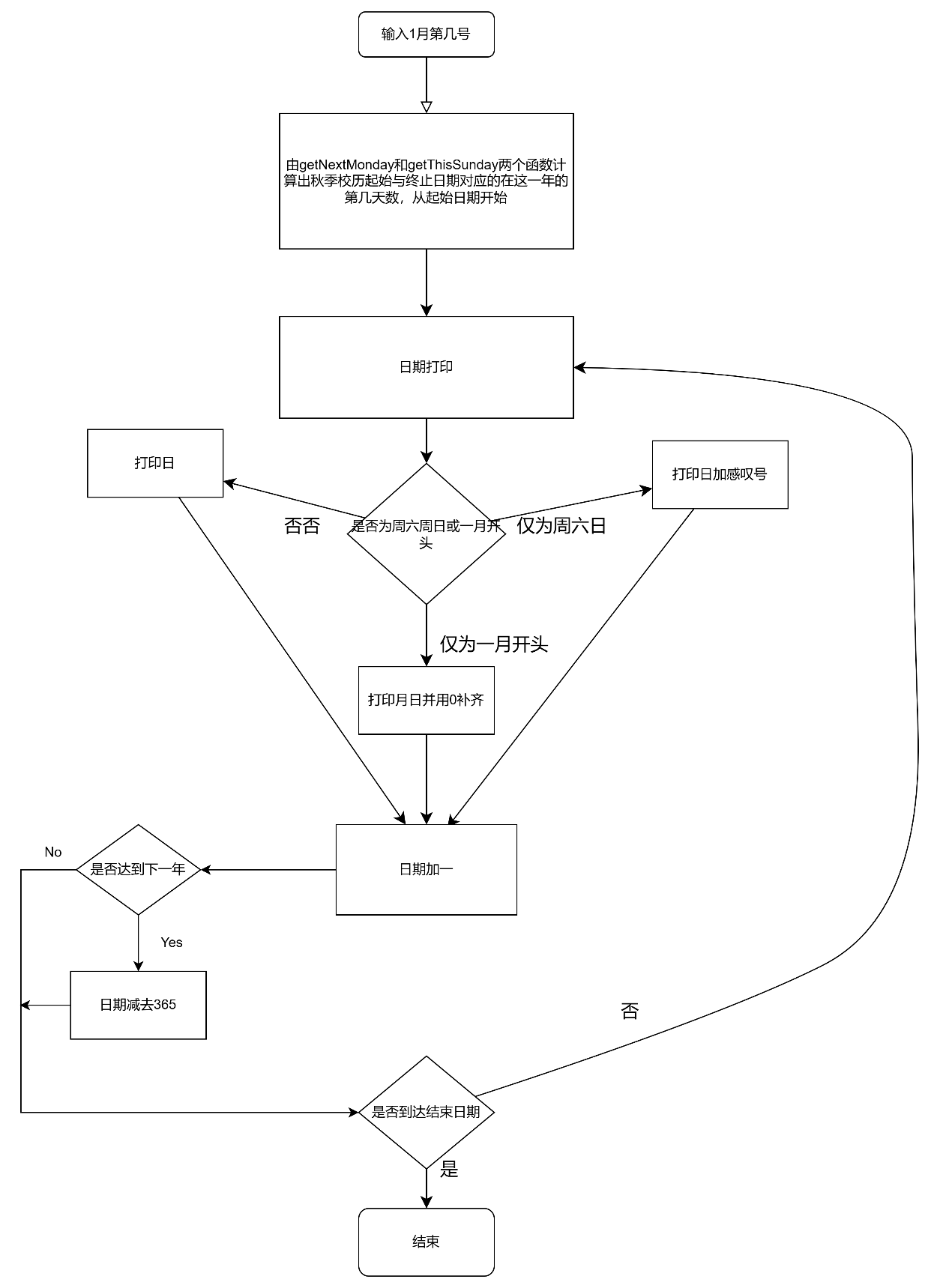
年1月的某日结束：用户输入1月份的某日作为

参考日，取该日所在周周日作为学期的结束。

– 更新相关函数，支持不同的年份的日期，计算

相关函数需要增加一个参数int year

## 实验步骤

关键代码：

int getDaySeq( int year, int month, int day)

{

if ( year == 2021 )

{

return ( getMonthLength( year, month ) + day );

}

if ( year == 2022 )

{

return ( getMonthLength( year, month ) + day + isLeapYear( year - 1 ) );

}

}

int getDaySeqOfWeek( int year, int daySeqOfYear)

{

if ( ( daySeqOfYear + 4 ) % 7 != 0 )

{

return (( daySeqOfYear + 4 )% 7 );

}

else

{

return 7;

}

}

int getNextMonday( int year, int daySeqOfYear)

{

if ( getDaySeqOfWeek( year, daySeqOfYear ) == 1 )

{

return daySeqOfYear;

}

else

{

while ( getDaySeqOfWeek( year, daySeqOfYear ) != 1 )

{

daySeqOfYear ++;

}

return daySeqOfYear;

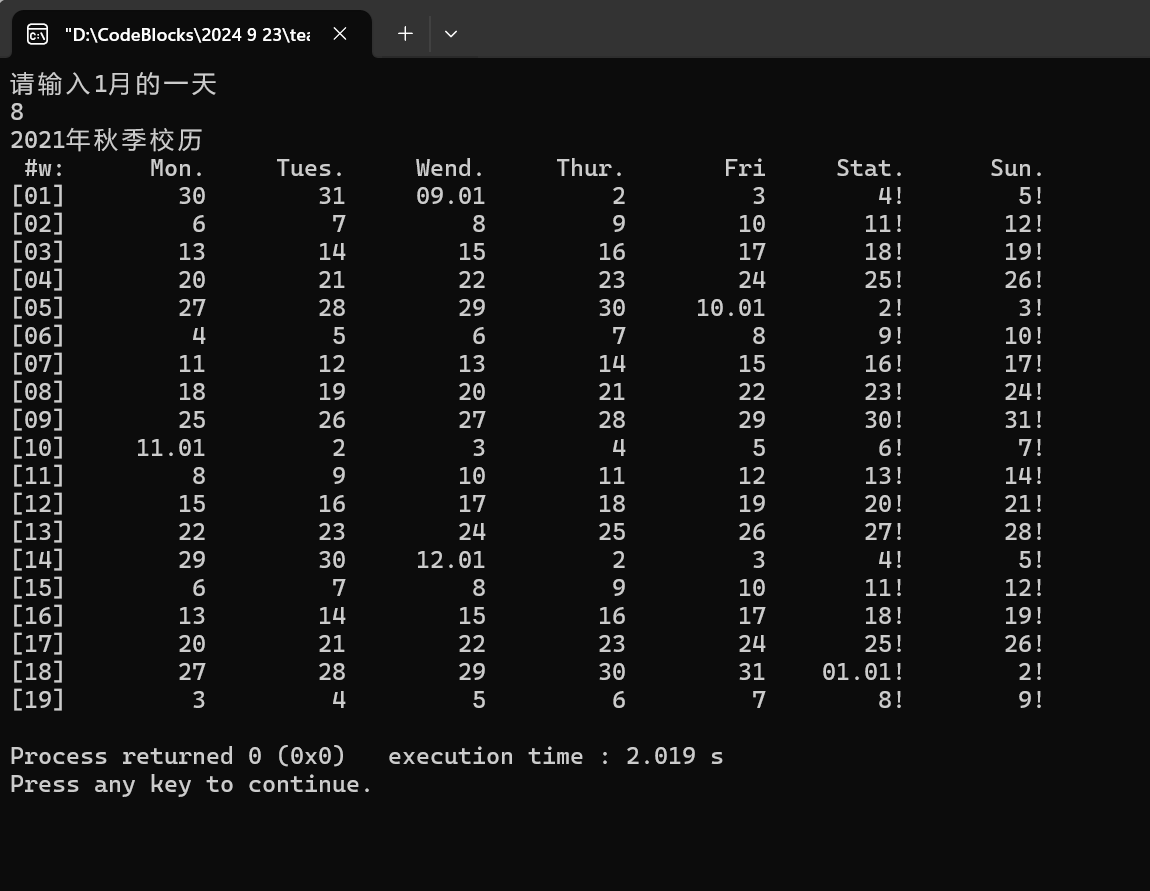
}

}

## 代码测试

### 测试点 “是否可以合适的开始”的测试结果

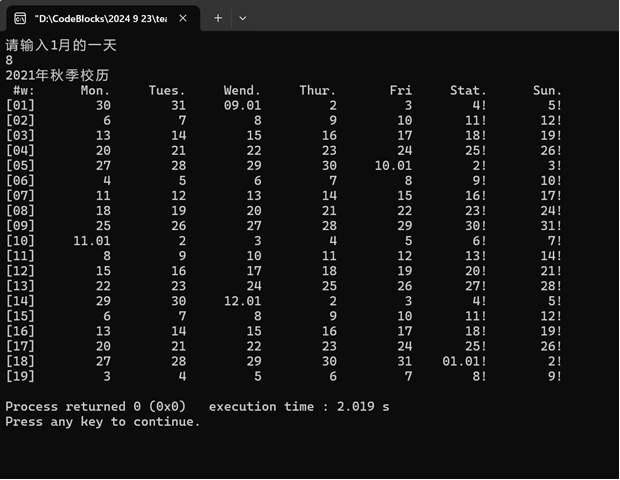
--->



预期：输入日期，从9.1那周周一开始的校历

结论：没问题

### 测试点 “是否能在周六周日后面加感叹号，每月1号改成0x.0x的形式”的测试结果

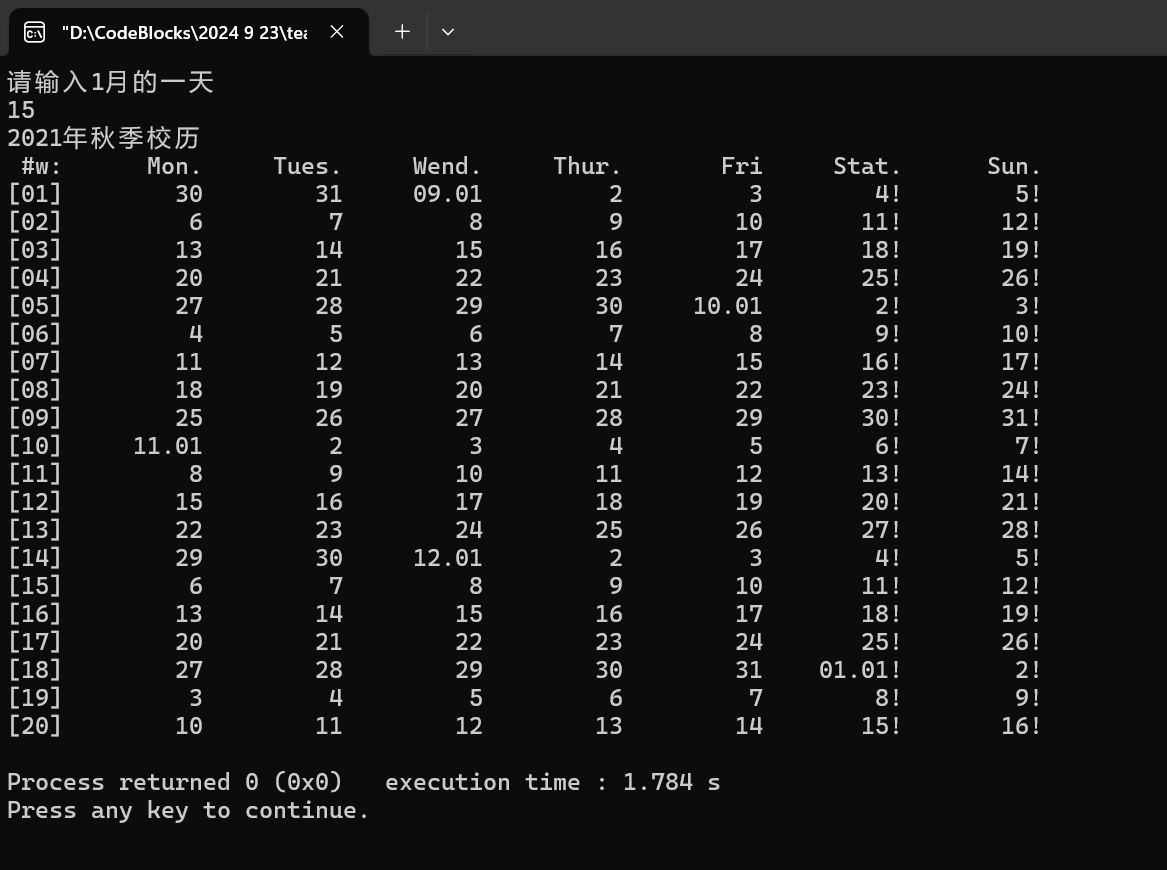


预期结果： 周六周日后面加感叹号，每月1号改成0x.0x的形式

结论：测试无问题

### 测试点 “能否换年的时候重新从1月开始”的测试结果

-->

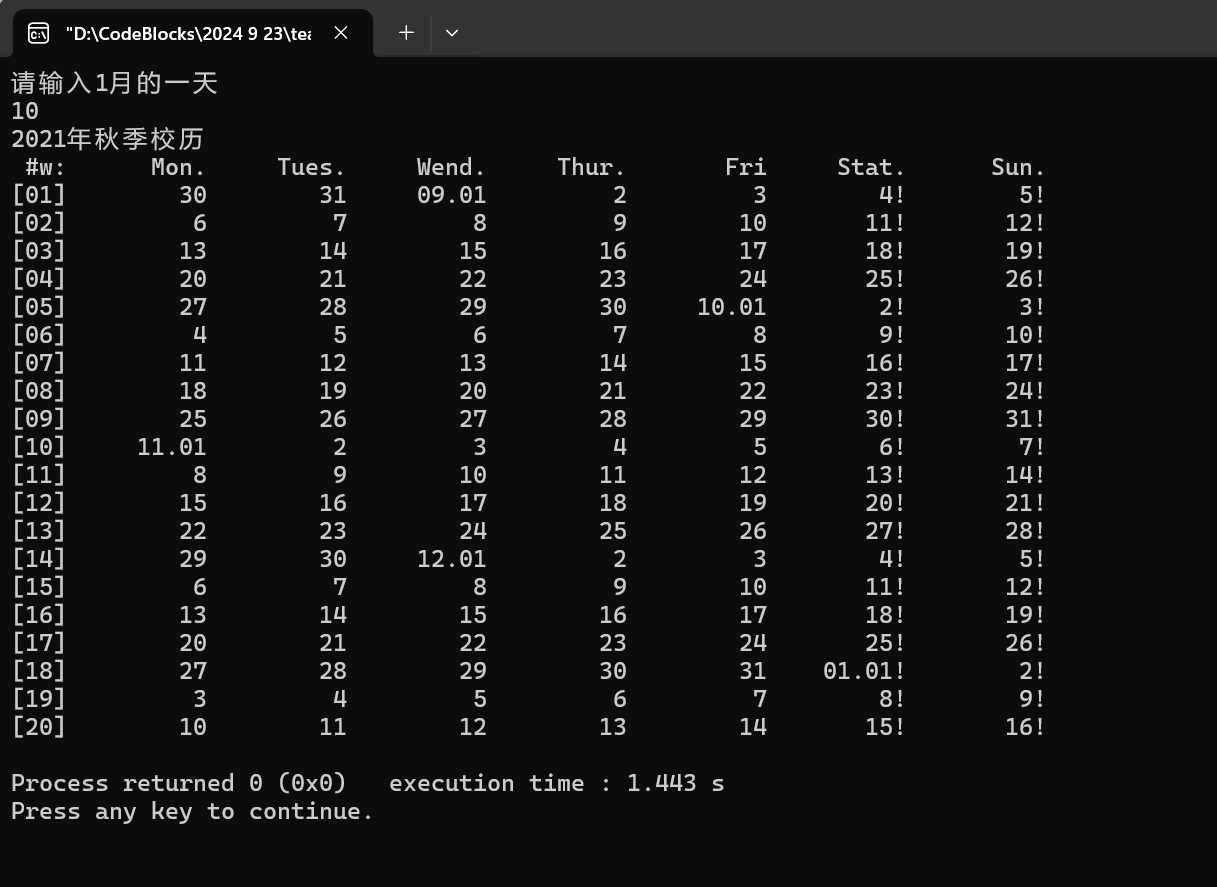


预期结果：在12月后从1月重新开始

结果：测试无问题

### 测试点 “是否输入日期的那周的周末作为终止日期”的测试结果

-->



预期结果：比如输入10，就在16号停止

结果：测试无问题

## 实验结论

代码达到功能目标

## 实验总结

跟上一题差不多，但是！需要特别细致，因为所有的函数都加了一个变量year，要一点一点分析year影响这个函数的什么，还要保证所有地方修改到位，花了很长时间，很复杂，还报错了好几次，比如年的天数减重复了什么的，真是困难，需要很细致

# 实验六

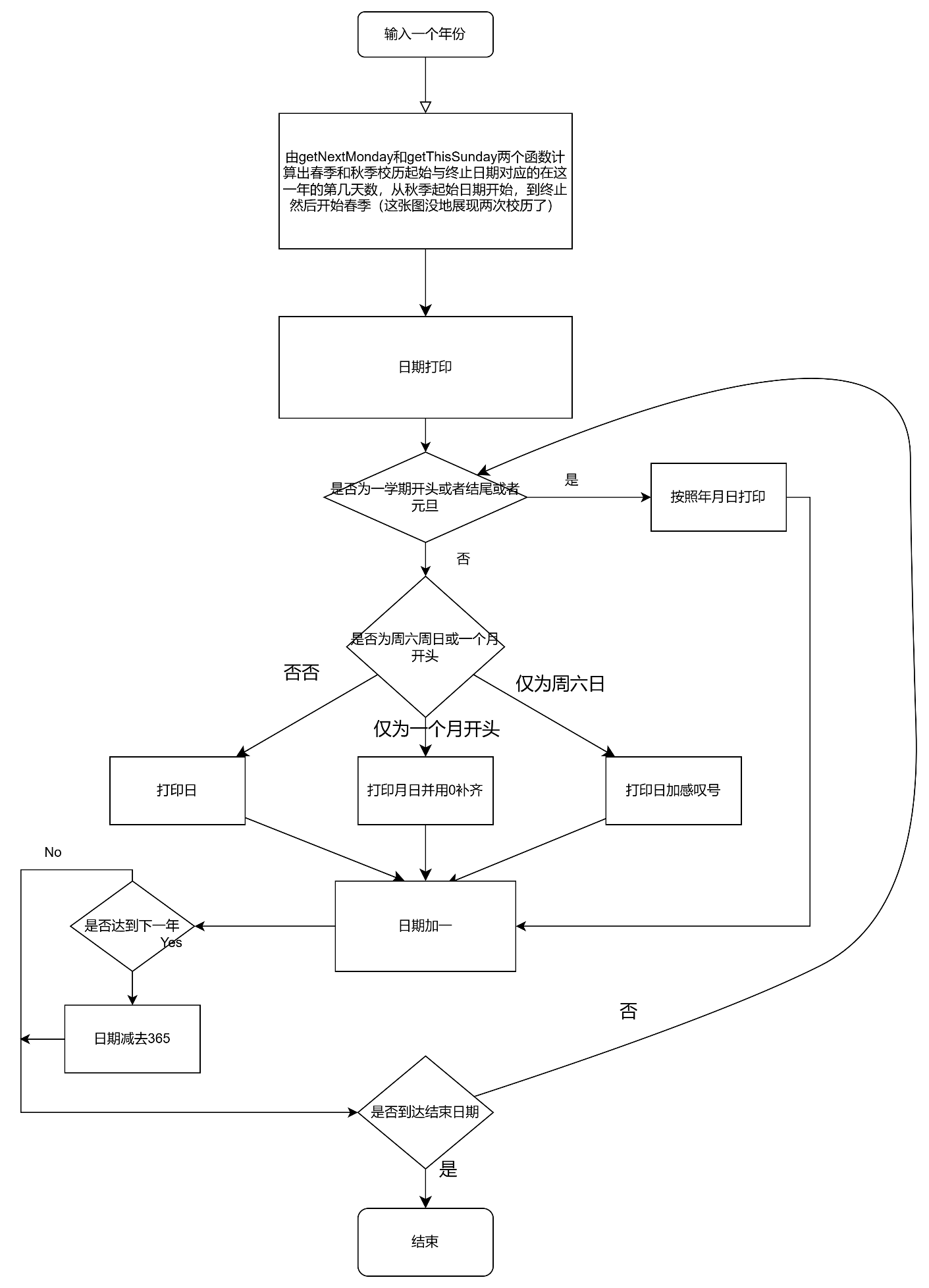
## 实验任务

用函数改写打印华中科技大学校历的程序，打印指定学年度的第一学期和第二学期的校历

– 约定秋季学期从当年9月4日所在周的周一开始，到第二年1月20日之前一周结束；约定春季学期从次年2月15日之后一周开始，7月第一周结束。

– 要求设计一个打印某周周历的函数，支持不同学期校历的打印需求；改进打印某日的函数，支持在校历首日和跨年日显示年月日信息。

## 实验步骤



关键代码：

void printOneDay( int year, int daySeqOfYear)

{

if ( (getMonth( year, daySeqOfYear) == 1 && getDay( year, daySeqOfYear) == 1 ) ||

daySeqOfYear == getThisMonday ( year, getDaySeq( year, 9, 4)) || daySeqOfYear == getThisSunday ( year, getDaySeq( year, 1,13)) ||

daySeqOfYear == getThisMonday ( year, getDaySeq( year, 2,22)) || daySeqOfYear == getThisSunday ( year, getDaySeq( year, 7, 1)))

{

if ( getDaySeqOfWeek( year, daySeqOfYear) == 6 || getDaySeqOfWeek( year, daySeqOfYear) == 7 )

{

printf (" %02d.%02d.%02d!", year % 100, getMonth( year, daySeqOfYear), getDay( year, daySeqOfYear));

}

else

{

printf (" %02d.%02d.%02d", year % 100, getMonth( year, daySeqOfYear ), getDay( year, daySeqOfYear ));

}

}

else

{

if ( getDay( year, daySeqOfYear) == 1 )

{

if ( getDaySeqOfWeek( year, daySeqOfYear) == 6 || getDaySeqOfWeek( year, daySeqOfYear) == 7 )

{

printf (" %02d.%02d!", getMonth( year, daySeqOfYear), getDay( year, daySeqOfYear));

}

else

{

printf (" %02d.%02d", getMonth( year, daySeqOfYear ), getDay( year, daySeqOfYear ));

}

}

else

{

if ( getDaySeqOfWeek( year, daySeqOfYear ) == 6 || getDaySeqOfWeek( year, daySeqOfYear ) == 7 )

{

printf ("%9d!", getDay( year, daySeqOfYear ));

}

else

{

printf ("%10d", getDay( year, daySeqOfYear ));

}

}

}

}

void printOneWeek( int currentYear, int daySeqOfWeek, int weekSeqOfSemester, int daySeqOfYear)

{

if ( daySeqOfWeek == 0 )

{

printf("[%02d]", weekSeqOfSemester );

}

// Print the month and day for the current day

if ( daySeqOfYear <= isLeapYear( currentYear ) )

{

printOneDay( currentYear, daySeqOfYear ) ;

}

else

{

printOneDay( currentYear + 1, daySeqOfYear ) ;

}

// After Sunday, print a new line

if ( daySeqOfWeek == 6 )

{

printf ("\n");

weekSeqOfSemester ++;

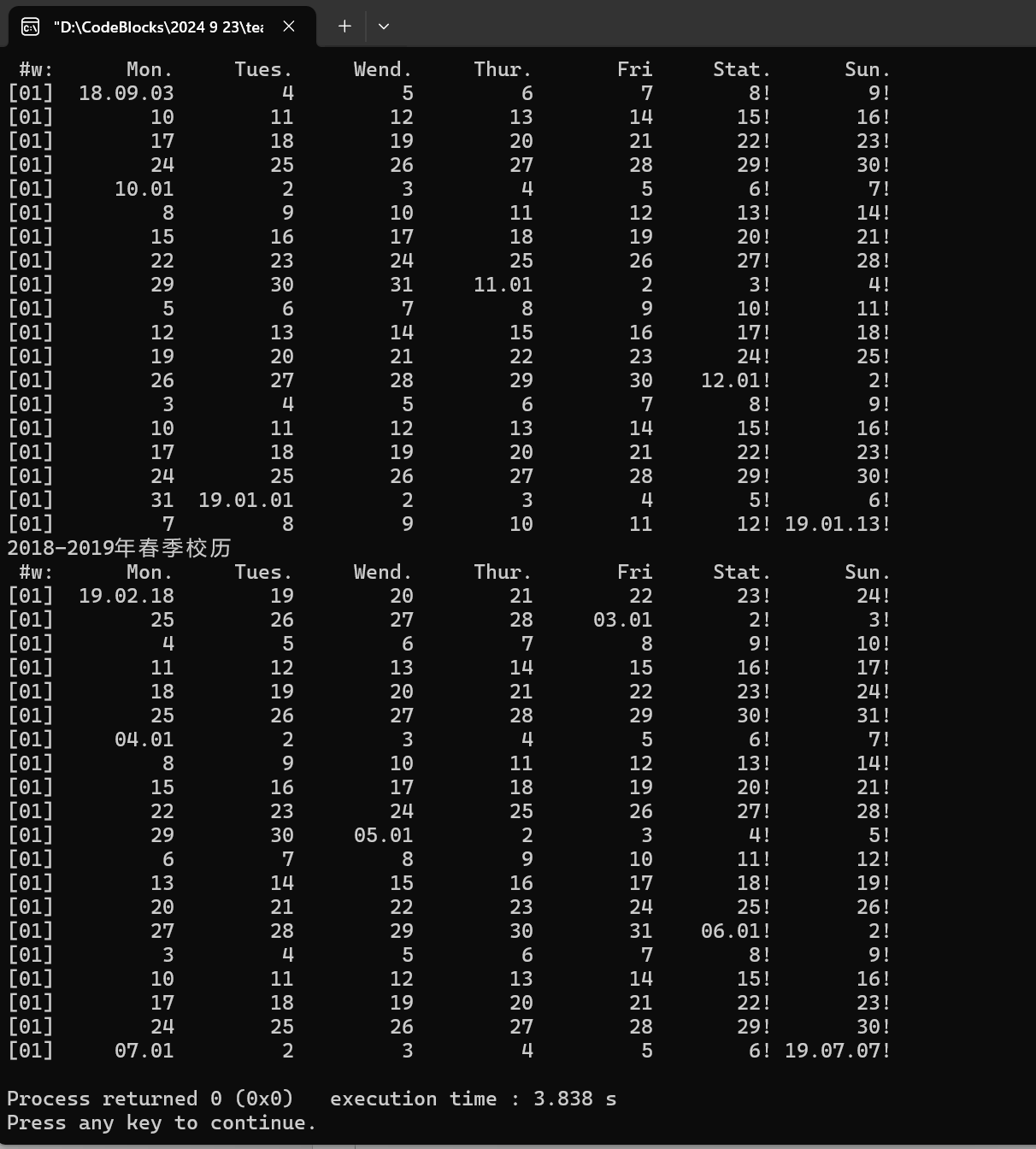
}

}

## 代码测试

### 测试点 “是否可以按照年份打印两个学期的校历”的测试结果

--->-->

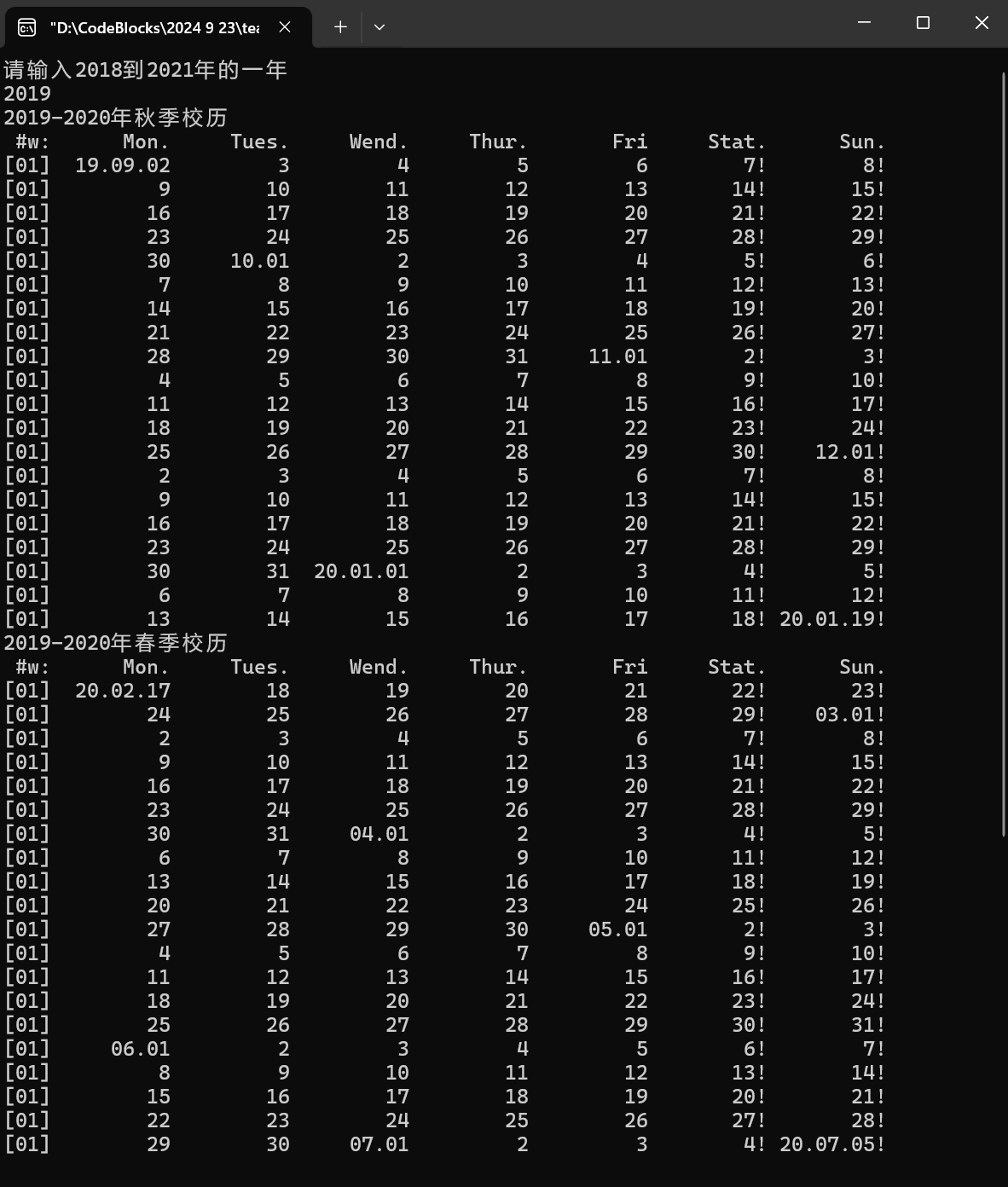


预期：输入年份，可以得到当学年两个学期的校历

结论：没问题

### 测试点 “是否能按规则进行春秋校历的开始与结束”的测试结果

-->

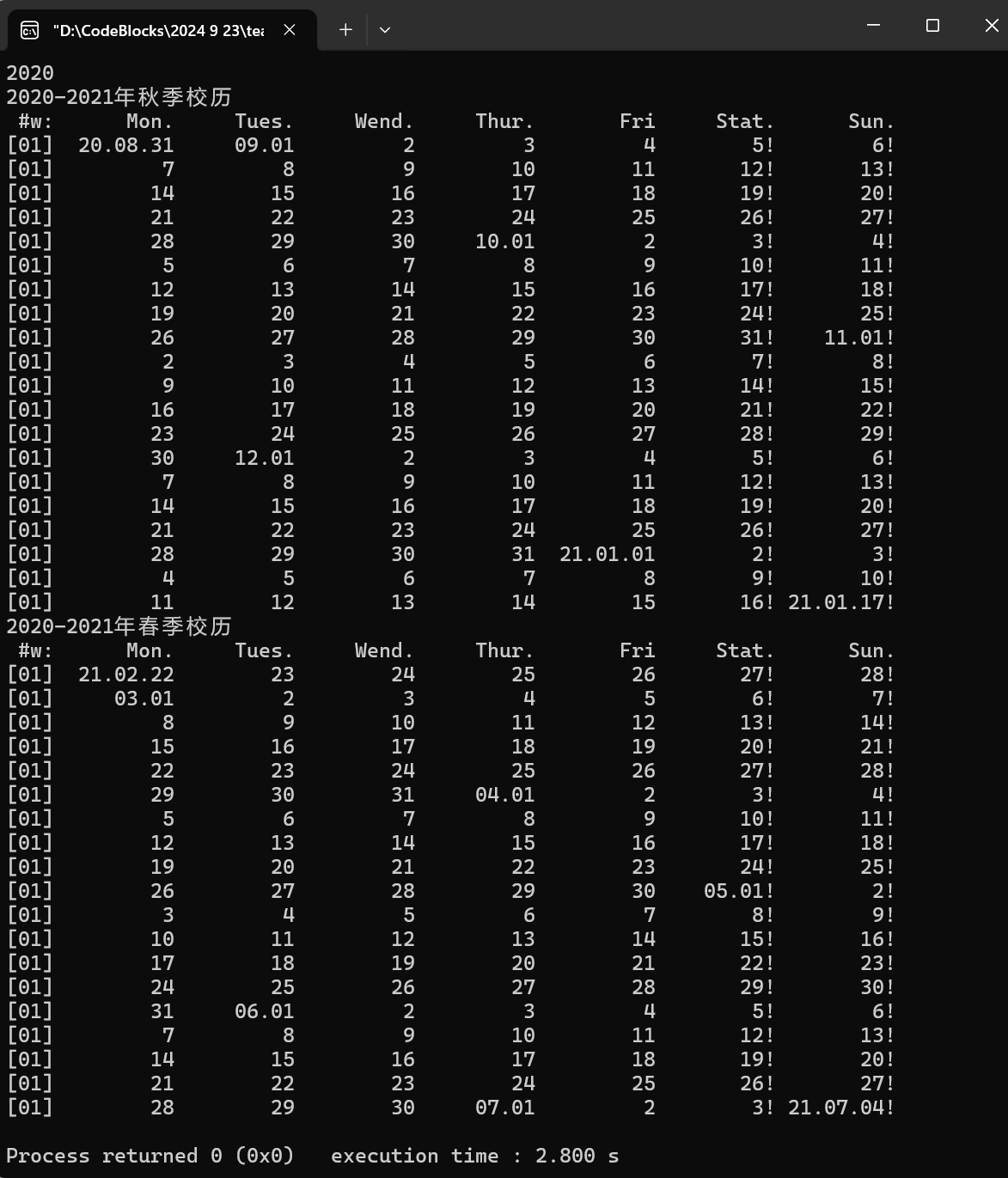


预期结果： 秋季学期从当年9月4日所在周的周一开始，到第二年1月20日之前一周结束；约定春季学期从次年2月15日之后一周开始，7月第一周结束

结论：测试无问题

### 测试点 “能否再校历开始结束和一年的第一天进行特殊表示”的测试结果

-->-->



预期结果： 能否再校历开始结束和一年的第一天输出年月日

结果：测试无问题

## 实验结论

代码达到功能目标

## 实验总结

跟上一题差不多，但是又新增了周历模块，还要从2018年开始，要有逻辑地改动原来代码，我先是把周历模块化出来，再把原本的一个学期的校历变成两个，再把一年改成多年，再把校历开始结束和元旦进行特殊表达。这道题逻辑性好的话，就可以一点一点过关斩将把它拿下。

# 实验八

## 实验任务

– 用数组改写打印华中科技大学校历的程序，具备打印校历的能力，比较多个年份的校历秋季校

历的首日，寻找开学最早的那一年

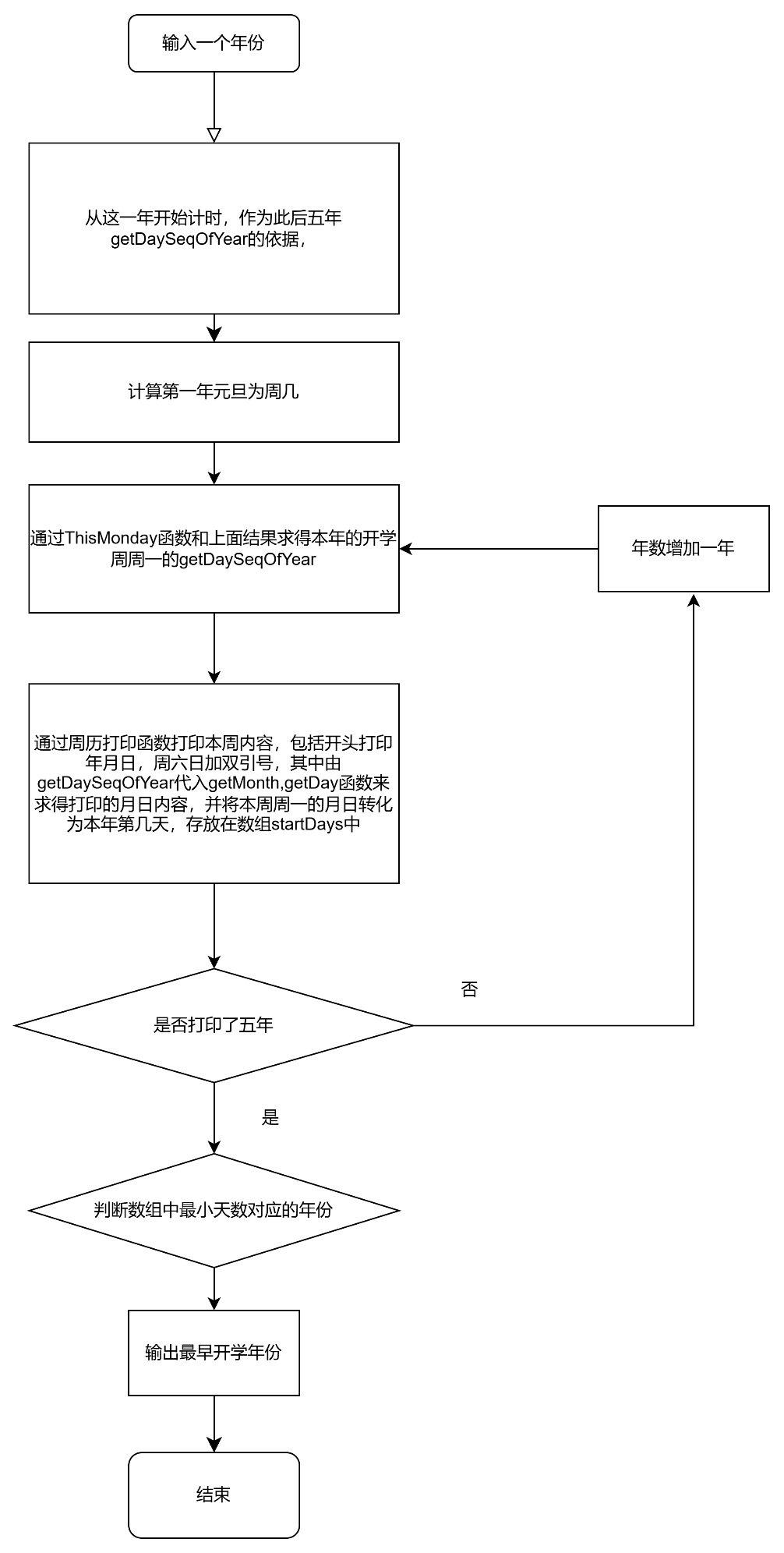
– 约定秋季学期从当年9月4日所在周的周一开始，到第二年1月20日之前一周结束；约定春季学

期从次年2月15日之后一周开始，7月第一周结束。

– 用两个只读的全局的一维数组记录不同月份的天数长度

– 基于数组改进现有的日历程序，改进计算日期的函数

## 实验步骤



关键代码：

int getThisMonday( int year, int daySeqOfYear, int inputYear)

{

if ( getDaySeqOfWeek( year, daySeqOfYear, inputYear ) == 1 )

{

return daySeqOfYear;

}

else

{

while ( getDaySeqOfWeek( year, daySeqOfYear, inputYear ) != 1 )

{

daySeqOfYear --;

}

return daySeqOfYear;

}

}

int isLeapYear( int year, int inputYear )

{

int length = 0;

for ( ; year >= inputYear; year -- )

{

if ( year % 4 != 0 )

{

length = length + 365;

}

else

{

length = length + 366;

}

}

return length;

}

int getDaySeq( int year, int month, int day, int inputYear)

{

if ( year == inputYear )

{

return ( getMonthLength( year, month ) + day );

}

if ( year > inputYear )

{

return ( getMonthLength( year, month ) + day + isLeapYear( year - 1, inputYear ) );

}

}

int getDaySeqOfWeek( int year, int daySeqOfYear, int inputYear)

{

int first;//这个是求输入年份的元旦前一天的周几

first = ( isLeapYear( 2019, inputYear - 1 ) + 1 ) % 7;

if ( ( daySeqOfYear + first ) % 7 != 0 )

{

return (( daySeqOfYear + first )% 7 );

}

else

{

return 7;

}

}

int getMonth ( int year, int dlay, int inputYear )

{

int common;

if ( year == inputYear )

{

common = dlay;

}

if ( year > inputYear )

{

common = dlay - isLeapYear( year - 1, inputYear ) ;

}

int month = 0;

if ( year % 4 != 0 )

{

for ( ; common > 0; month ++ )

{

common = common - Month\_NORMAL\_YEAR [month];

}

return month;

}

else

{

for ( ; common > 0; month ++ )

{

common = common - Month\_LEAP\_YEAR [month];

}

return month;

}

}

## 代码测试

### 测试点 “是否为可以输出从输入年份到后五年开学周历”的测试结果

预期：输出从输入年份连续五年秋季开学周历

--> -->

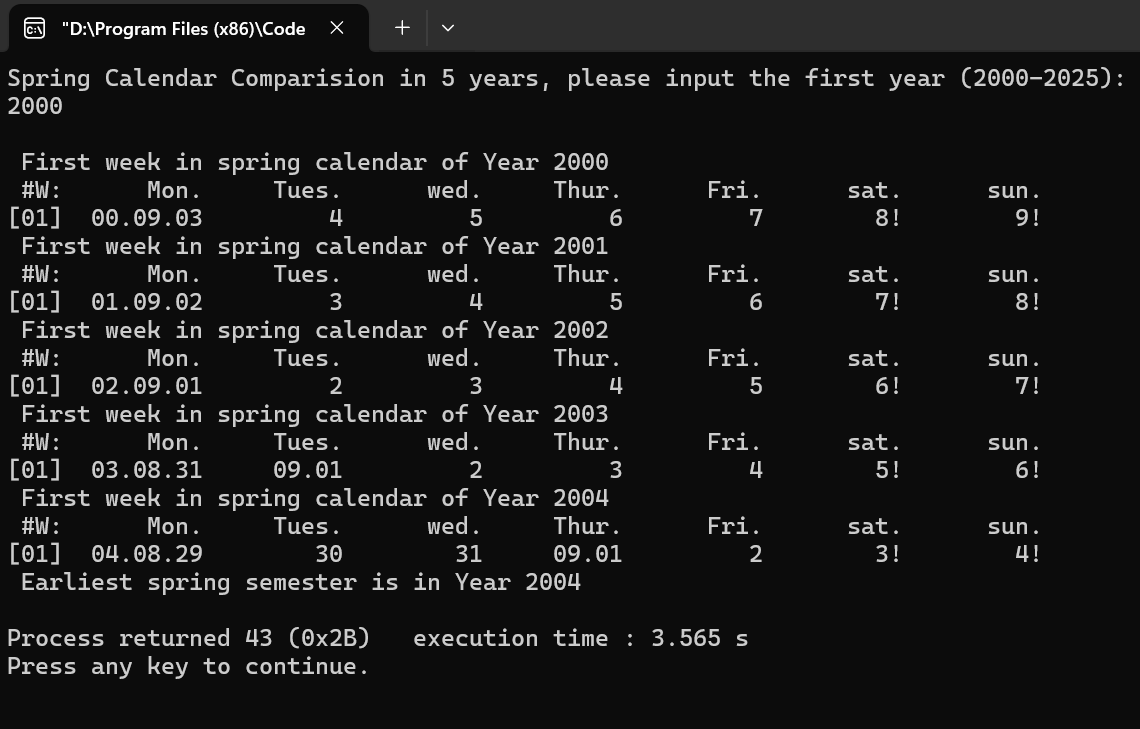


测试无问题

### 测试点 “是否可以按照规范输出9.4所在周周一，并且周一用年月日，周六日加感叹号输出”的测试结果

预期：可以按照规范输出9.4所在周周一，并且周一用年月日，周六日加感叹号输出

-->



结论：测试无问题

### 测试点 “是否可以按照规则，打印五年里最早开学的年份”的测试结果

预期结果：打印年份确实是最早开学的年份

-->



结论：测试没问题

## 实验结论

代码达到功能目标

## 实验总结

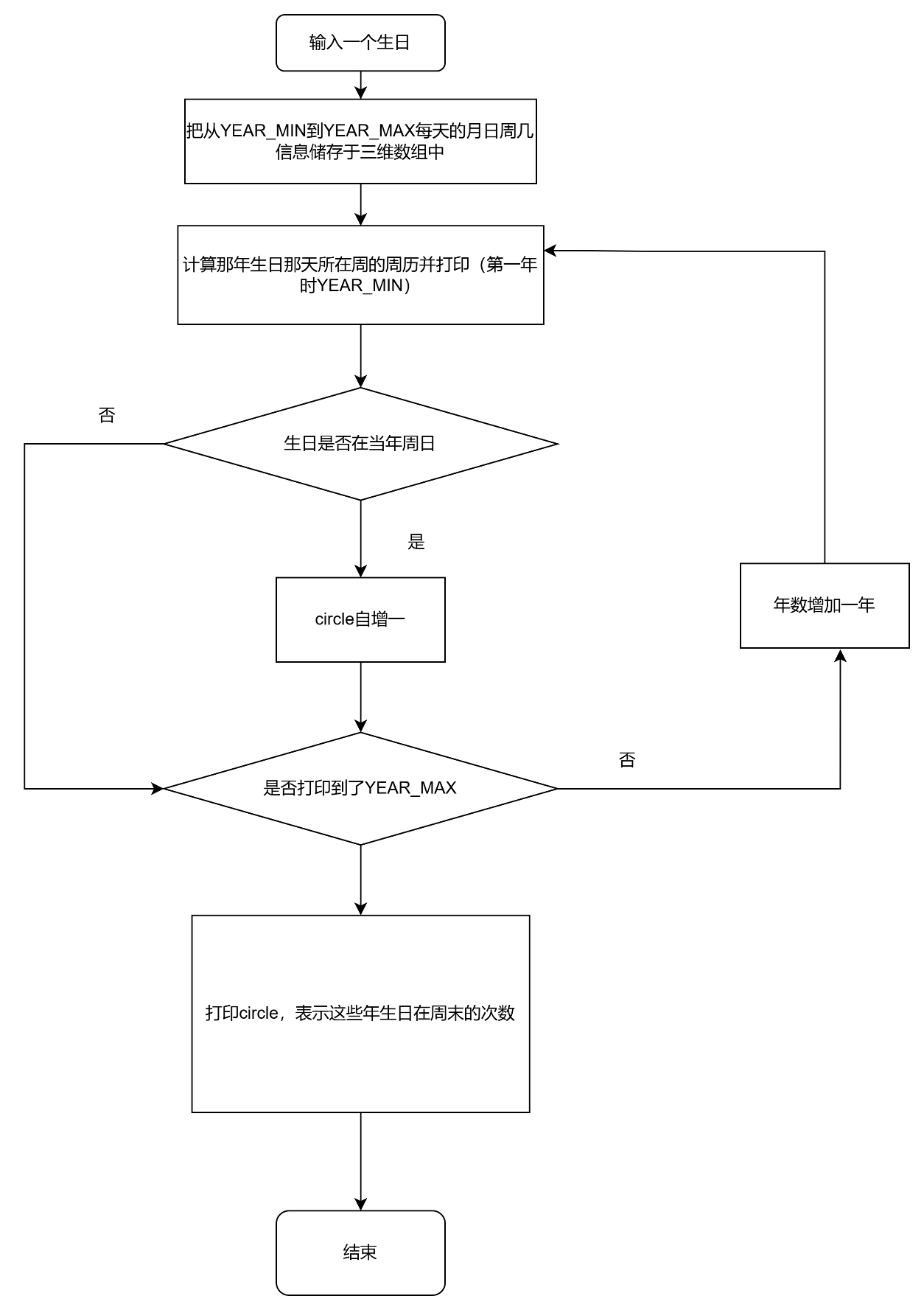
这个题和前面发生了质的改变，变化就出现在1.年份为任意，这改变了日期到序列数那部分的函数，2.月份数组的加入，导致月份到序列数，序列数到日期的函数都变化了，3.题目给的main函数变了，等于说这个题虽然和前面情景很像，但是是大相径庭的，我于是重写了一遍，上下兼顾，左顾右盼，生怕出了什么小细节的错误，万幸我成功了。再写一遍，感觉写的更加利索简洁，更加直接高效，感受到了自己代码技术和思维的变化。写到最后出现正确结果时真的很骄傲

# 实验九

## 实验任务

**用三维数组记录多年的日期数据，查找某人的生日，并打印该生日所在周的周历，计算其在周末过生日的次数**

## 实验步骤



关键代码：

for(i = 0 ; i < YEAR\_MAX - YEAR\_MIN + 1; i ++ )

{

year = Years[i];

if ( year % 4 == 0 )

{

yearLength = 366;

}

else

{

yearLength = 365;

}

for ( j = 0 ; j < yearLength; j ++ )

{

Days[i][j][0] = getMonth( year, j + 1 + isLeapYear( year - 1, inputYear), inputYear);

Days[i][j][1] = getDay( year, j + 1 + isLeapYear( year - 1, inputYear), inputYear);

Days[i][j][2] = j + 1 + isLeapYear( year - 1, inputYear);

Days[i][j][3] = getDaySeqOfWeek( year, j + 1 + isLeapYear( year - 1, inputYear), inputYear);

}

}

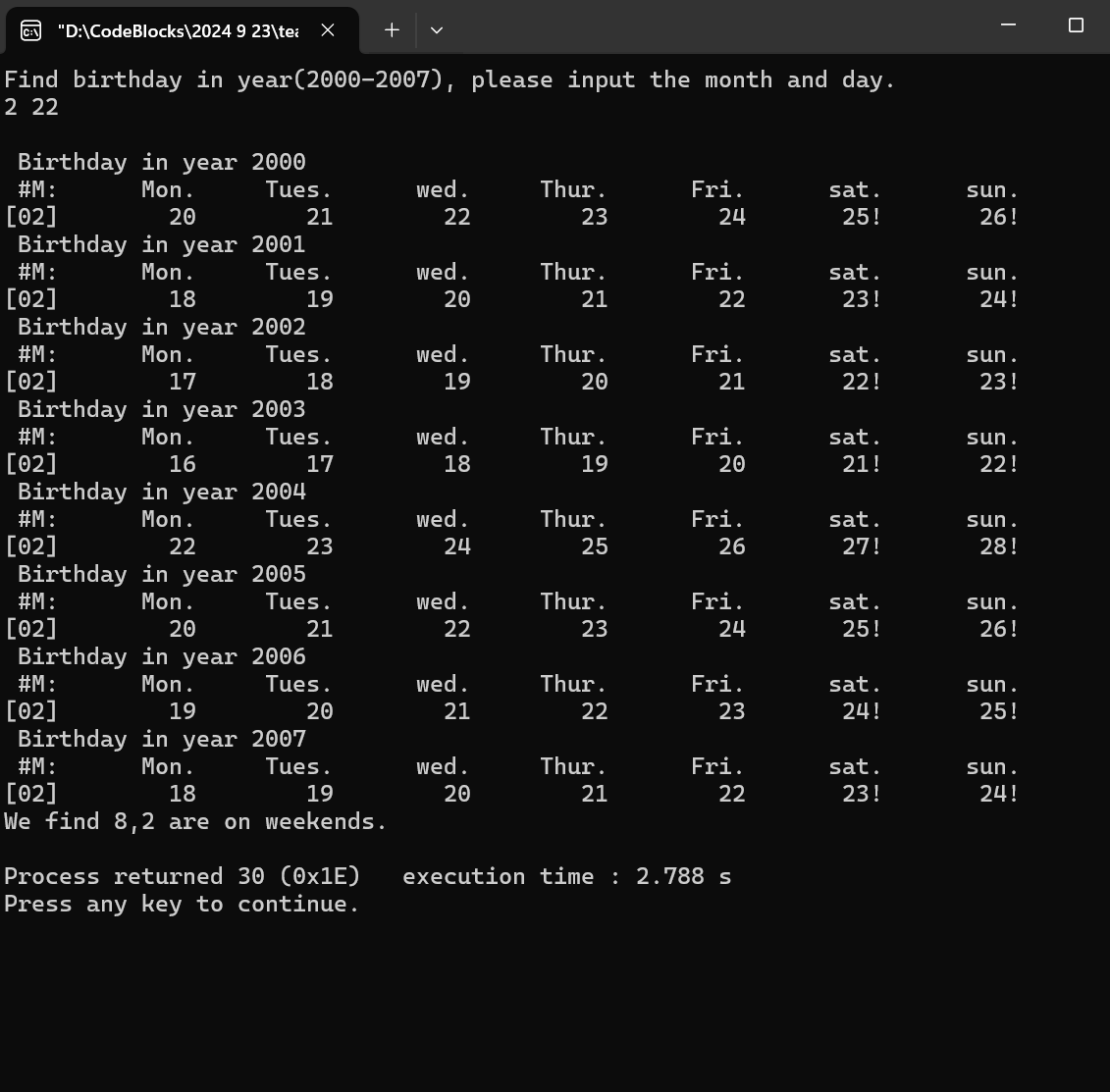
}

## 代码测试

### 测试点 “是否可以输出生日对应的周历”的测试结果

预期：输出了生日所在的周历

--> -->

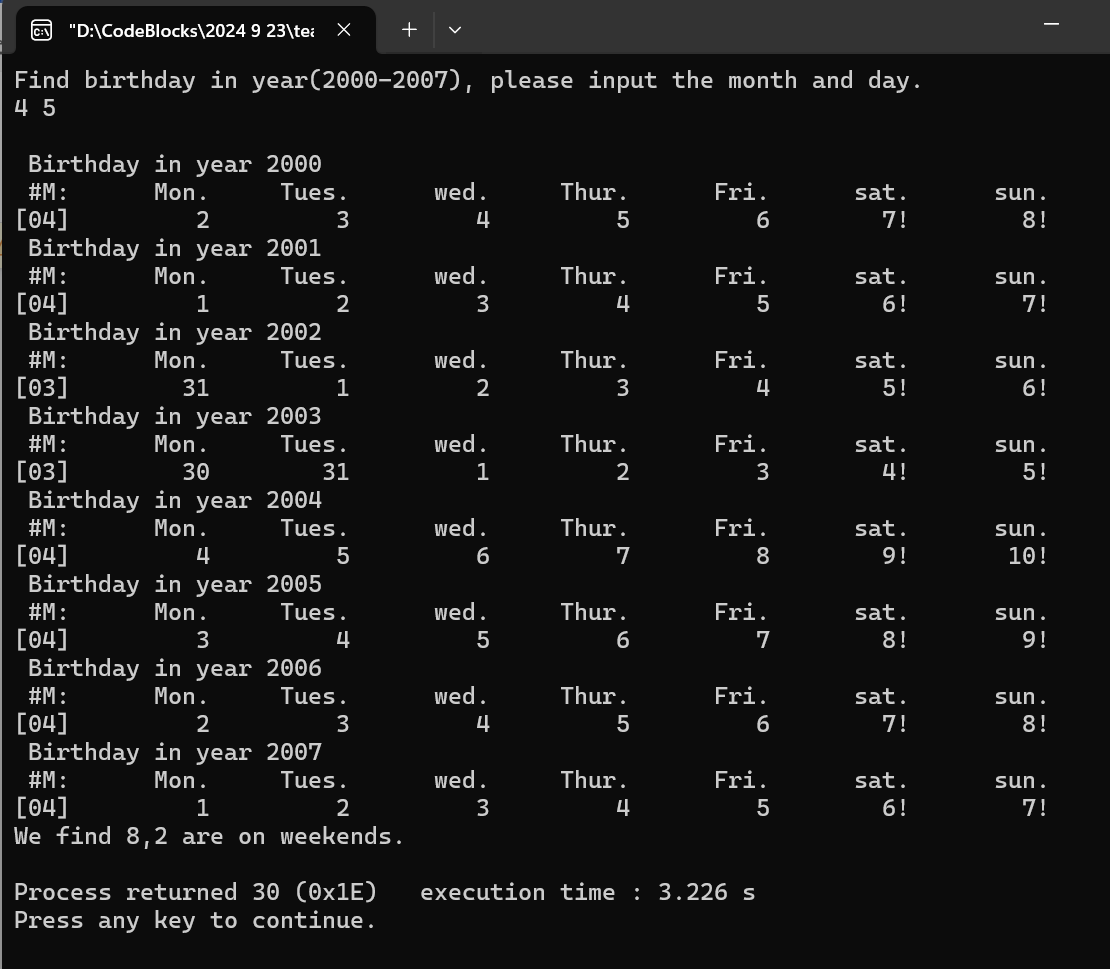


测试无问题

### 测试点 “是否可以正确打印有多少次生日在周末”的测试结果

预期：正确打印在周末生日的次数

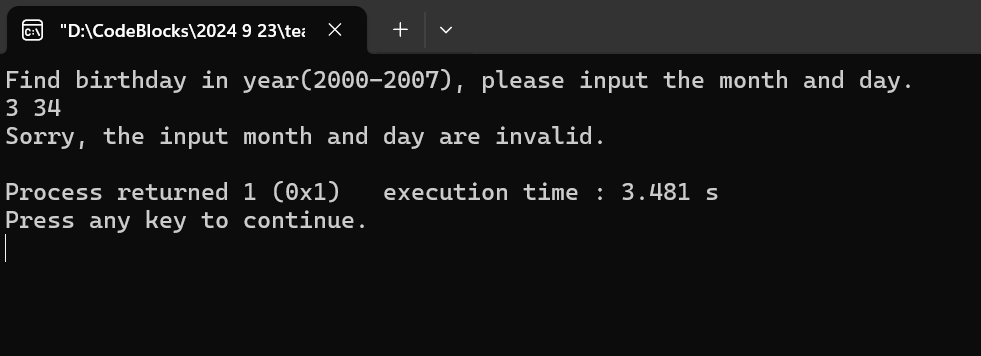
-->



### 测试点 “是否可以辨别不存在日期”的测试结果

预期：在不存在时可以输出提示话语

-->



## 实验结论

代码达到功能目标

## 实验总结

这个三维数组的加入让我很难办，其实它没有那么难，但是我输进去就是运行不了，排查一遍又一遍，这个函数看着就是没问题，苦恼好久，再main函数里又看，发现把这个函数放在对Years数组的赋值之前了，挪了一下位置，豁然开朗了。

# 实验十、寻找生日（结构体）

## 实验任务

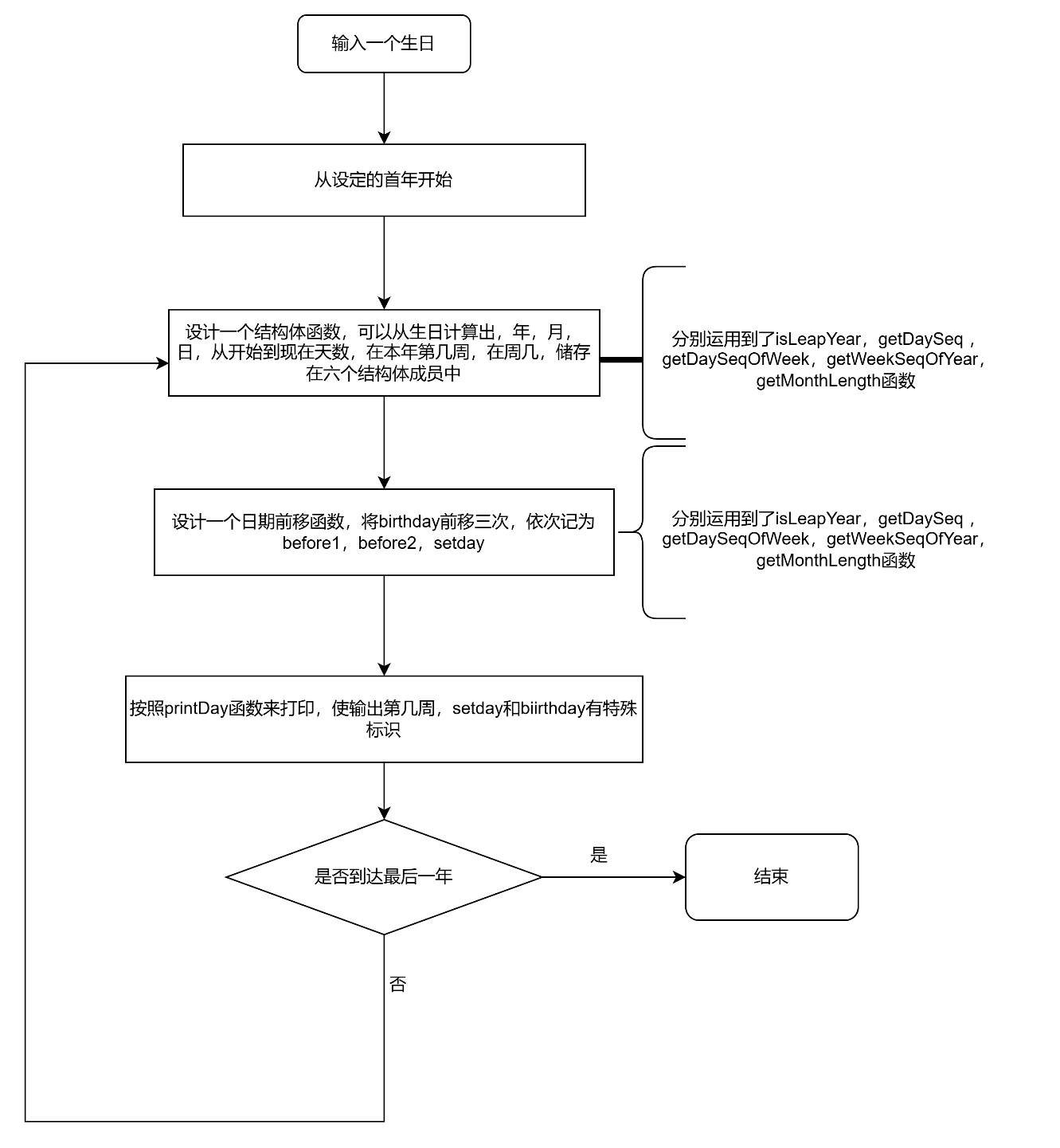
假定某生日趴需要三天时间准备，输入某人的生日，通过日期偏移计算获得前

三天并打印相关周历

– 用日期结构体记录

单一日期的所有数据属性：

## 实验步骤



关键代码：

for ( i = 0; i < YEAR\_MAX - YEAR\_MIN + 1; i++ )

{

printf("\n%s%s%d\n"," ","Birthday in year ", Years[i]);

printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.","Tues.","wed.","Thur.","Fri.","sat.","sun.");

Day birthday = setDay ( Years[i], inputMonth, inputDay, inputYear);

Day before1 = moveDay ( birthday, inputYear);

Day before2 = moveDay ( before1, inputYear);

Day setday = moveDay ( before2, inputYear);

printfDay ( birthday, before1, before2, setday);

}

int getWeekSeqOfYear( int year, int daySeqOfYear, int inputYear)

{

return ( daySeqOfYear + 5 - getDaySeqOfWeek( year, daySeqOfYear, inputYear)) / 7 + 1 - isLeapYear( year - 1, inputYear ) / 7;

}

Day setDay ( int year, int month, int day, int inputYear)

{

Day day1;

day1.year = year;

day1.daySeq = getDaySeq( year, month, day, inputYear);

day1.month = month;

day1.day = day;

day1.weekSeq = getWeekSeqOfYear( year, day1.daySeq, inputYear);

day1.weekDay = getDaySeqOfWeek( year, day1.daySeq, inputYear);

return day1;

}

Day moveDay ( Day day1, int inputYear )

{

Day day2;

if ( day1.month == 1 && day1.day == 1 )

{

day2.year = day1.year - 1;

}

else

{

day2.year = day1.year;

}

day2.daySeq = day1.daySeq - 1;

day2.month = getMonth ( day2.year, day2.daySeq, inputYear );

day2.day = getDay ( day2.year, day2.daySeq, inputYear );

day2.weekSeq = getWeekSeqOfYear( day2.year, day2.daySeq, inputYear);

day2.weekDay = getDaySeqOfWeek( day2.year, day2.daySeq, inputYear);

return day2;

}

void printfDay ( Day birthday, Day before1, Day before2, Day setDay)

{

if ( setDay.weekDay == 1 )

{

printf ("[%02d] %02d.%02d.%02d%10d%10d %02d.%02d\*\n", setDay.weekSeq, setDay.year % 100, setDay.month,

setDay.day, before2.day, before1.day, birthday.month, birthday.day);

}

if ( setDay.weekDay == 2 )

{

printf ("[%02d] %2d.%02d.%02d%10d%10d %02d.%02d\*\n", setDay.weekSeq, setDay.year % 100, setDay.month,

setDay.day, before2.day, before1.day, birthday.month, birthday.day);

}

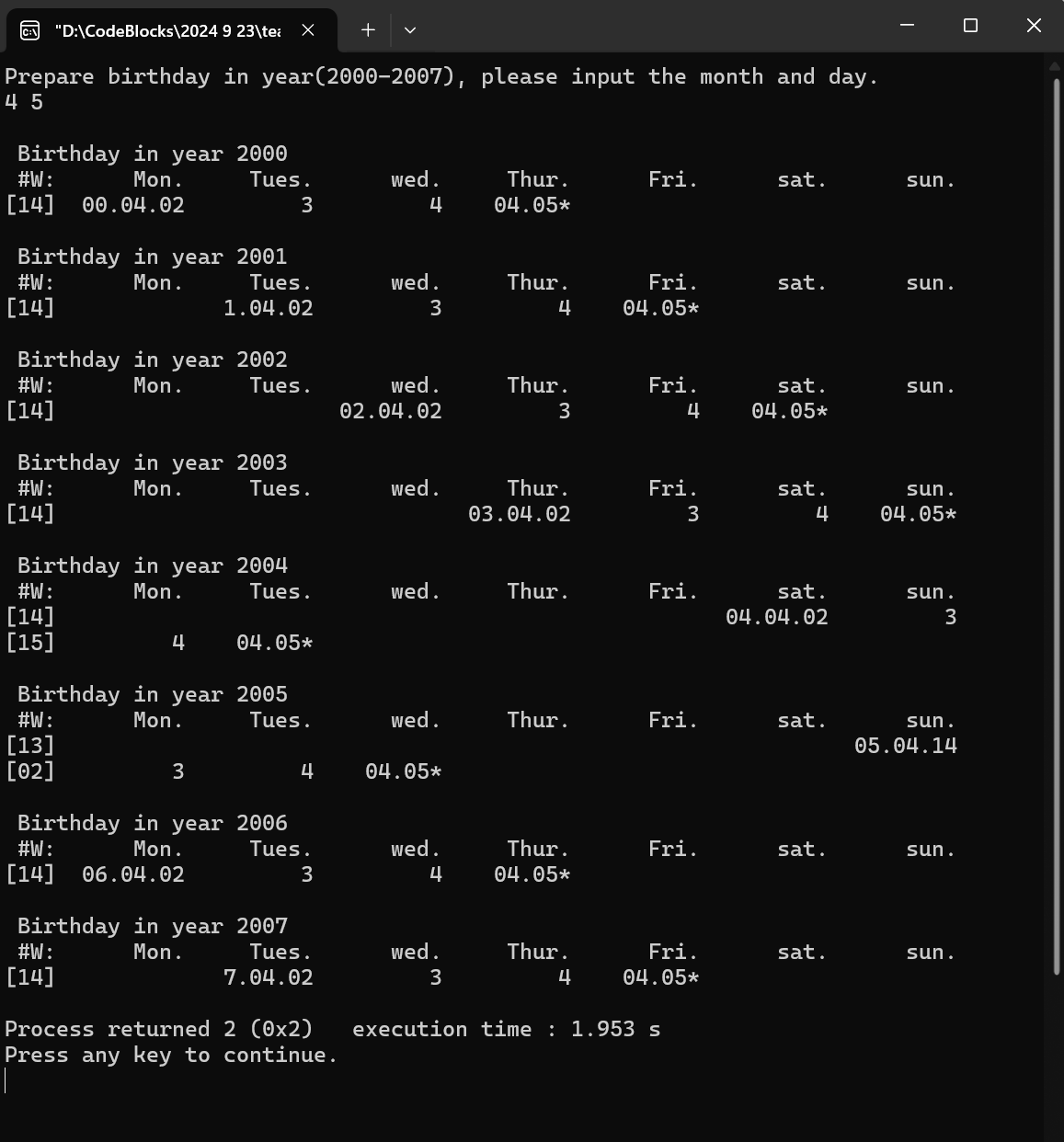
## 代码测试

### 测试点 “是否可以按照基本格式打印”的测试结果

预期：如果是，则可以1.从开始年到结束年2.只打印四天，且第一第四天特殊表示3.可以打印第几周

4.如果跨周了可以打印两周

--> -->

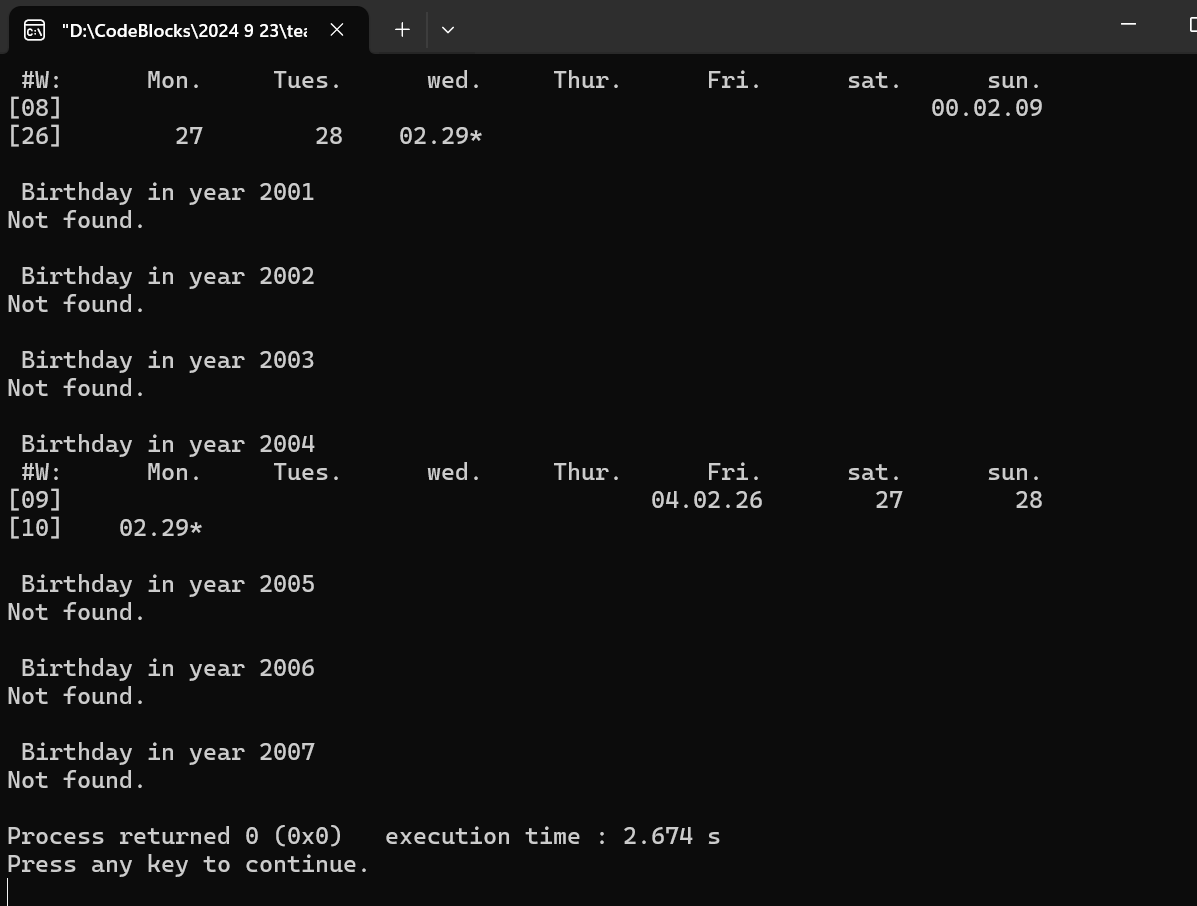


测试无问题

### 测试点 “是否可以将2.29特殊表示”的测试结果

预期：可以在不是闰年的时候只输出not found

-->



结果：符合预期

## 实验结论

代码达到功能目标

## 实验总结

这个也是，由于改成了结构体，我需要把main函数重新写一遍，但是我并不是死板的按照ppt写，而是在我理解之上自己创建了三个必备结构体函数，然后用我的思维写了整道题，除了日期计算的函数用的是之前的，其他的整个贯通自己编写了一遍，真正感受到了结构体的方便快捷，清晰易懂，这也是最后一道题了，整个写下来，收获颇丰。

# 本课程学习总结

整个学期学下来，就像这系列题，一开始学的东西很简单，但是应对一些比较复杂的东西就会比较繁杂；后来学的东西就更加高深更加深入，但是会写起来比之前简单的多，有点人生哲学了。

另外就是我个人，一开始学C语言特比特别痛苦，因为我对这种全新的，之前丝毫没有接触过的代码的世界非常非常陌生，我敲键盘都不熟练，然后课程缩短又赶得紧，常常是我听了课一知半解，再看看书还是不透彻，然后再去写题。这学是一回事，做又是另外一回事。我这个人特别犟，一个题过不去我就特别特别难受，但是我上头了之后眼力不好，把那个过不去的代码从头到尾从尾到头一点点看，还是看不出来，特别挫败。但是我知道，这是初学者的必经之路，我上大学后，已经从那个对高中东西了如指掌的诸葛亮，变成重新打怪升级，还是很不适应的，觉得写不出来特别挫败。

但是神奇的是，渐渐的，我学了数组，学了指针，学了结构体，再回去看之前的题，顿时觉得小菜一碟，这个时候心中非常非常自豪，我觉得我成长了，我挺过来了。

人生也是这样，有时候走到死胡同，就像是一个代码一直不对但是就是不知道哪里错了，可能再耐心一点，或者再经历多一点，回头看，死胡同也变成了阳关大道。

# 附录

1. 计算周几实验
2. **int** main(**void**)
3. {
5. **int** month,day,integer1;
7. puts("请输入月与日的数字，中间空一格");
9. scanf("%d %d",&month,&day);
11. **if** ( month == 1)
12. {
13. integer1 = day % 7;
14. }
15. **if** ( month == 2)
16. {
17. integer1 = (day + 31) % 7;
18. }
19. **if** ( month == 3)
20. {
21. integer1 = (day + 31 + 29) % 7;
22. }
23. **if** ( month == 4)
24. {
25. integer1 = (day + 31 + 29 + 31) % 7;
26. }
27. **if** ( month == 5)
28. {
29. integer1 = (day + 31 + 29 + 31 + 30) % 7;
30. }
31. **if** ( month == 6)
32. {
33. integer1 = (day + 31 + 29 + 31 + 30 + 31) % 7;
34. }
35. **if** ( month == 7)
36. {
37. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30) % 7;
38. }
39. **if** ( month == 8)
40. {
41. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31) % 7;
42. }
43. **if** ( month == 9)
44. {
45. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31) % 7;
46. }
47. **if** ( month == 10)
48. {
49. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30) % 7;
50. }
51. **if** ( month == 11)
52. {
53. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31) % 7;
54. }
55. **if** ( month == 12)
56. {
57. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30) % 7;
58. }
60. **if** ( integer1 == 0)
61. {
62. integer1 = integer1+7;
63. }
65. printf ( "%d月%d号是周%d", month, day, integer1 );
67. }

二、计算月历实验

(1) main. c

1. int main(void)
2. {
4. int month,day,integer1,integer2,num;
6. day = 1;
8. num = 1;
10. printf ("Month Calendar of Year 2024, please input month(1-12):\n");
12. scanf ("%d", & month);

15. if (month == 1)
16. {
17. integer1 = day % 7;
18. }
19. if (month == 2)
20. {
21. integer1 = (day + 31) % 7;
22. }
23. if (month == 3)
24. {
25. integer1 = (day + 31 + 29) % 7;
26. }
27. if (month == 4)
28. {
29. integer1 = (day + 31 + 29 + 31) % 7;
30. }
31. if (month == 5)
32. {
33. integer1 = (day + 31 + 29 + 31 + 30) % 7;
34. }
35. if (month == 6)
36. {
37. integer1 = (day + 31 + 29 + 31 + 30 + 31) % 7;
38. }
39. if (month == 7)
40. {
41. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30) % 7;
42. }
43. if (month == 8)
44. {
45. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31) % 7;
46. }
47. if (month == 9)
48. {
49. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31) % 7;
50. }
51. if (month == 10)
52. {
53. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30) % 7;
54. }
55. if (month == 11)
56. {
57. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31) % 7;
58. }
59. if (month == 12)
60. {
61. integer1 = (day + 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30) % 7;
62. }
64. if ( integer1 == 0)
65. {
66. integer1 = integer1+7;
67. }
69. integer2 = integer1;

72. //先在这加上那个周一到周日
73. printf ("%10s%10s%10s%10s%10s%10s%10s\n",
75. "Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
76. //先判断月份
77. if ( month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12 )
78. {
79. while ( ( integer1 - 1 ) != 0 )
80. {
81. printf("          ");
83. integer1 = integer1 - 1;
84. }
85. while ( num <= 31 )
86. {
87. if ( (num + integer2 - 1 ) % 7 != 0 )
88. {
90. printf( "%10d",num );
92. num = num + 1;
93. }
94. else
95. {
96. printf ( "%10d\n",num );
98. num = num + 1;
99. }
100. }
101. }
102. if ( month == 4 || month == 6 || month == 9 || month == 11 )
103. {
104. while ( ( integer1 - 1 ) != 0 )
105. {
106. printf("          ");
108. integer1 = integer1 - 1;
109. }
110. while ( num <= 30 )
111. {
112. if ( (num + integer2 - 1 ) % 7 != 0 )
113. {
115. printf( "%10d",num );
117. num = num + 1;
118. }
119. else
120. {
121. printf ( "%10d\n",num );
123. num = num + 1;
124. }
125. }
126. }
127. if ( month == 2 )
128. {
129. while ( ( integer1 - 1 ) != 0 )
130. {
131. printf("          ");
133. integer1 = integer1 - 1;
134. }
135. while ( num <= 29 )
136. {
137. if ( (num + integer2 - 1 ) % 7 != 0 )
138. {
140. printf( "%10d",num );
142. num = num + 1;
143. }
144. else
145. {
146. printf ( "%10d\n",num );
148. num = num + 1;
149. }
150. }
151. }
152. }

三、周历问题

1. #include <stdio.h>
2. #include <stdlib.h>
4. void PrintMonthDay ( int data );
6. int main(void)
7. {
8. int week;
9. int day;
10. printf ("请输入一个数字，代表2021年第几周\n");
11. scanf ("%d", &week );
12. printf ("%10s%10s%10s%10s%10s%10s%10s%10s\n",
14. "#w:","Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
15. if ( week > 1 && week < 53 )
16. {
17. day = week \* 7 - 3;
18. int integer1 = 0;//用来遍历一周
19. printf ("       %02d:", week );
20. for ( ; integer1 < 7; integer1 ++ )
21. {
22. PrintMonthDay ( day + integer1 );
23. }
24. }
25. else if ( week == 1 )
26. {
27. printf ("%10s%10s%10s%10s%10s%10s%10s%10s\n",
29. "01:"," "," "," ","","01.01","01.02","01.03");
30. }
31. else if ( week == 53 )
32. {
33. printf ("%10s%10s%10s%10s%10s%10s%10s%10s\n",
35. "53:","12.27","12.28","12.29","12.30","12.31", " "," ");
36. }
37. }
39. void PrintMonthDay ( int data )
40. {
41. int month;
42. int day;
43. if ( data <= 31 )
44. {
45. month = 1;
46. day = data;
47. printf ("     %02d.%02d", month, day );
48. return;
49. }
50. if ( data <= 31 + 28 )
51. {
52. month = 2;
53. day = data - 31;
54. printf ("     %02d.%02d", month, day );
55. return;
56. }
57. if ( data <= 31 + 28 + 31 )
58. {
59. month = 3;
60. day = data - 31 - 28;
61. printf ("     %02d.%02d", month, day );
62. return;
63. }
64. if ( data <= 31 + 28 + 31 + 30 )
65. {
66. month = 4;
67. day = data - 31 - 28 - 31;
68. printf ("     %02d.%02d", month, day );
69. return;
70. }
71. if ( data <= 31 + 28 + 31 + 30 + 31)
72. {
73. month = 5;
74. day = data - 31 - 28 - 31 -30;
75. printf ("     %02d.%02d", month, day );
76. return;
77. }
78. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
79. {
80. month = 6;
81. day = data - 31 - 28 - 31 - 30 - 31;
82. printf ("     %02d.%02d", month, day );
83. return;
84. }
85. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
86. {
87. month = 7;
88. day = data - 31 - 28 - 31 - 30 - 31 - 30;
89. printf ("     %02d.%02d", month, day );
90. return;
91. }
92. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
93. {
94. month = 8;
95. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
96. printf ("     %02d.%02d", month, day );
97. return;
98. }
99. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
100. {
101. month = 9;
102. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
103. printf ("     %02d.%02d", month, day );
104. return;
105. }
106. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
107. {
108. month = 10;
109. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
110. printf ("     %02d.%02d", month, day );
111. return;
112. }
113. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 )
114. {
115. month = 11;
116. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
117. printf ("     %02d.%02d", month, day );
118. return;
119. }
120. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
121. {
122. month = 12;
123. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
124. printf ("     %02d.%02d", month, day );
125. return;
126. }
127. }

四、春季校历2020年问题

1. #include <stdio.h>
2. #include <stdlib.h>
4. void PrintMonthDay ( int data );
6. int main(void)
7. {
8. int weekday;
9. int day;
10. int week;
11. int integerweek = 1;//这是打印时输出的学期内周次
12. printf ("请输入一个数字，代表2月几日\n");
13. scanf ("%d", &weekday );
14. if (( weekday + 31 + 2 ) % 7 == 1 || ( weekday + 31 + 2 ) % 7 == 0 )
15. {
16. week = ( weekday + 31 ) / 7 + 1;
17. }
18. else
19. {
20. week = ( weekday + 31 ) / 7 + 2;
21. }
22. printf ("%10s%10s%10s%10s%10s%10s%10s%10s\n",
24. "#w:","Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
25. for ( ; week <= 26; week ++ )
26. {
27. day = week \* 7 - 1;
28. int integer1 = 0;//用来遍历一周
29. printf ("       %02d:", integerweek );
30. integerweek ++;
31. for ( ; integer1 < 7; integer1 ++ )
32. {
33. PrintMonthDay ( day + integer1 );
34. }
35. printf ("\n");
36. }
37. }
39. void PrintMonthDay ( int data )
40. {
41. int month;
42. int day;
43. if ( data <= 31 + 29 )
44. {
45. month = 2;
46. day = data - 31;
47. printf ("     %02d.%02d", month, day );
48. return;
49. }
50. if ( data <= 31 + 29 + 31 )
51. {
52. month = 3;
53. day = data - 31 - 29;
54. printf ("     %02d.%02d", month, day );
55. return;
56. }
57. if ( data <= 31 + 29 + 31 + 30 )
58. {
59. month = 4;
60. day = data - 31 - 29 - 31;
61. printf ("     %02d.%02d", month, day );
62. return;
63. }
64. if ( data <= 31 + 29 + 31 + 30 + 31)
65. {
66. month = 5;
67. day = data - 31 - 29 - 31 -30;
68. printf ("     %02d.%02d", month, day );
69. return;
70. }
71. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 )
72. {
73. month = 6;
74. day = data - 31 - 29 - 31 - 30 - 31;
75. printf ("     %02d.%02d", month, day );
76. return;
77. }
78. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 )
79. {
80. month = 7;
81. day = data - 31 - 29 - 31 - 30 - 31 - 30;
82. printf ("     %02d.%02d", month, day );
83. return;
84. }
    1. 2021春季校历函数版
85. #include <stdio.h>
86. #include <stdlib.h>
87. // Functionsabout monthand day
88. int getMonthLength( int month);
89. int getDaySeq(int month, int day);
90. // Functions for propertiesofoneday
91. int getMonth( int daySeqOfYear);
92. int getDay( int daySeqOfYear);
93. int getDaySeqOfWeek(int daySeqOfYear);
94. // Functions for daymovement calculation
95. int getNextMonday( int daySeqOfYear);
96. int getThisSunday( int daySeqOfYear);
97. void printOneDay( int daySeqOfYear);
98. // Calculate the first day and last day in the spring semester
99. int main(void)
100. {
101. int referenceDay;
102. printf ("请输入2月的一天\n");
103. scanf ("%d", &referenceDay );
104. printf ("2021年春季校历\n");
105. printf ("%4s%10s%10s%10s%10s%10s%10s%10s\n",
106. "#w:","Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
107. int sStartSeqOfYear = getNextMonday( getDaySeq(2, referenceDay) );
108. int sEndSeqOfYear = getThisSunday( getDaySeq(7, 1) ) ;
109. // Navigate every day in this semester
110. int daySeqOfYear, daySeqOfWeek, weekSeqOfSemester ;
111. for ( daySeqOfYear = sStartSeqOfYear, daySeqOfWeek = 0, weekSeqOfSemester = 1;
112. daySeqOfYear <= sEndSeqOfYear;
113. daySeqOfYear ++, daySeqOfWeek ++, daySeqOfWeek %= 7 )
114. {
115. // Before Monday, print the week sequence
116. if ( daySeqOfWeek == 0 )
117. {
118. printf("[%02d]", weekSeqOfSemester );
119. }
120. // Print the month and day for the current day
121. printOneDay( daySeqOfYear ) ;
122. // After Sunday, print a new line
123. if ( daySeqOfWeek == 6 )
124. {
125. printf ("\n");
126. weekSeqOfSemester ++;
127. }
128. }
129. return 0;
130. }
131. int getMonthLength( int month)
132. {
133. if ( month == 1 )
134. {
135. return 0;
136. }
137. if ( month == 2 )
138. {
139. return 31;
140. }
141. if ( month == 3 )
142. {
143. return ( 31 + 28 );
144. }
145. if ( month == 4 )
146. {
147. return ( 31 + 28 + 31 );
148. }
149. if ( month == 5 )
150. {
151. return ( 31 + 28 + 31 + 30 );
152. }
153. if ( month == 6 )
154. {
155. return ( 31 + 28 + 31 + 30 + 31 );
156. }
157. if ( month == 7 )
158. {
159. return ( 31 + 28 + 31 + 30 + 31 + 30 );
160. }
161. if ( month == 8 )
162. {
163. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 );
164. }
165. if ( month == 9 )
166. {
167. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 );
168. }
169. if ( month == 10 )
170. {
171. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 );
172. }
173. if ( month == 11 )
174. {
175. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 );
176. }
177. if ( month == 12 )
178. {
179. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 );
180. }
181. }
182. int getDaySeq(int month, int day)
183. {
184. return ( getMonthLength( month ) + day ) ;
185. }
186. int getDaySeqOfWeek(int daySeqOfYear)
187. {
188. if ( ( daySeqOfYear + 4 ) % 7 != 0 )
189. {
190. return (( daySeqOfYear + 4 )% 7 );
191. }
192. else
193. {
194. return 7;
195. }
196. }
197. int getNextMonday( int daySeqOfYear)
198. {
199. if ( getDaySeqOfWeek( daySeqOfYear ) == 1 )
200. {
201. return daySeqOfYear;
202. }
203. else
204. {
205. while ( getDaySeqOfWeek( daySeqOfYear ) != 1 )
206. {
207. daySeqOfYear ++;
208. }
209. return daySeqOfYear;
210. }
211. }
212. int getThisSunday( int daySeqOfYear)
213. {
214. if ( getDaySeqOfWeek( daySeqOfYear ) == 7 )
215. {
216. return daySeqOfYear;
217. }
218. else
219. {
220. while ( getDaySeqOfWeek( daySeqOfYear ) != 7 )
221. {
222. daySeqOfYear ++;
223. }
224. return daySeqOfYear;
225. }
226. }
227. int getMonth( int daySeqOfYear)
228. {
229. int data = daySeqOfYear;
230. int month;
231. int day;
232. if ( data <= 31 )
233. {
234. month = 1;
235. day = data;
236. return month;
237. }
238. if ( data <= 31 + 28 )
239. {
240. month = 2;
241. day = data - 31;
242. return month;
243. }
244. if ( data <= 31 + 28 + 31 )
245. {
246. month = 3;
247. day = data - 31 - 28;
248. return month;
249. }
250. if ( data <= 31 + 28 + 31 + 30 )
251. {
252. month = 4;
253. day = data - 31 - 28 - 31;
254. return month;
255. }
256. if ( data <= 31 + 28 + 31 + 30 + 31)
257. {
258. month = 5;
259. day = data - 31 - 28 - 31 -30;
260. return month;
261. }
262. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
263. {
264. month = 6;
265. day = data - 31 - 28 - 31 - 30 - 31;
266. return month;
267. }
268. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
269. {
270. month = 7;
271. day = data - 31 - 28 - 31 - 30 - 31 - 30;
272. return month;
273. }
274. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
275. {
276. month = 8;
277. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
278. return month;
279. }
280. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
281. {
282. month = 9;
283. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
284. return month;
285. }
286. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
287. {
288. month = 10;
289. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
290. return month;
291. }
292. }
293. int getDay( int daySeqOfYear)
294. {
295. int data = daySeqOfYear;
296. int month;
297. int day;
298. if ( data <= 31 )
299. {
300. month = 1;
301. day = data;
302. return day;
303. }
304. if ( data <= 31 + 28 )
305. {
306. month = 2;
307. day = data - 31;
308. return day;
309. }
310. if ( data <= 31 + 28 + 31 )
311. {
312. month = 3;
313. day = data - 31 - 28;
314. return day;
315. }
316. if ( data <= 31 + 28 + 31 + 30 )
317. {
318. month = 4;
319. day = data - 31 - 28 - 31;
320. return day;
321. }
322. if ( data <= 31 + 28 + 31 + 30 + 31)
323. {
324. month = 5;
325. day = data - 31 - 28 - 31 -30;
326. return day;
327. }
328. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
329. {
330. month = 6;
331. day = data - 31 - 28 - 31 - 30 - 31;
332. return day;
333. }
334. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
335. {
336. month = 7;
337. day = data - 31 - 28 - 31 - 30 - 31 - 30;
338. return day;
339. }
340. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
341. {
342. month = 8;
343. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
344. return day;
345. }
346. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
347. {
348. month = 9;
349. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
350. return day;
351. }
352. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
353. {
354. month = 10;
355. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
356. return day;
357. }
358. }
359. void printOneDay( int daySeqOfYear)
360. {
361. if ( getDay(daySeqOfYear) == 1 )
362. {
363. if ( getDaySeqOfWeek(daySeqOfYear) == 6 || getDaySeqOfWeek(daySeqOfYear) == 7 )
364. {
365. printf (" %02d.%02d!", getMonth(daySeqOfYear), getDay(daySeqOfYear));
366. }
367. else
368. {
369. printf (" %02d.%02d", getMonth(daySeqOfYear), getDay(daySeqOfYear));
370. }
371. }
372. else
373. {
374. if ( getDaySeqOfWeek(daySeqOfYear) == 6 || getDaySeqOfWeek(daySeqOfYear) == 7 )
375. {
376. printf ("%9d!", getDay(daySeqOfYear));
377. }
378. else
379. {
380. printf ("%10d", getDay(daySeqOfYear));
381. }
382. }
383. }

（六） 2021秋季校历

1. #include <stdio.h>
2. #include <stdlib.h>
3. int isLeapYear( int year );
4. // Functionsabout monthand day
5. int getMonthLength( int year, int month);
6. int getDaySeq( int year, int month, int day);
7. // Functions for propertiesofoneday
8. int getMonth( int year, int daySeqOfYear);
9. int getDay( int year, int daySeqOfYear);
10. int getDaySeqOfWeek( int year, int daySeqOfYear);
11. // Functions for daymovement calculation
12. int getNextMonday( int year, int daySeqOfYear);
13. int getThisSunday( int year, int daySeqOfYear);
14. int getThisMonday( int year, int daySeqOfYear);
15. void printOneDay( int year, int daySeqOfYear);
16. // Calculate the first day and last day in the spring semester
17. int main(void)
18. {
19. int referenceDay;
20. int currentYear = 2021;
21. printf ("请输入1月的一天\n");
22. scanf ("%d", &referenceDay );
23. printf ("2021年秋季校历\n");
24. printf ("%4s%10s%10s%10s%10s%10s%10s%10s\n",
25. "#w:","Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
26. int sStartSeqOfYear = getThisMonday( currentYear, getDaySeq( currentYear, 9, 1) );
27. int sEndSeqOfYear = getThisSunday( currentYear + 1, getDaySeq( currentYear + 1, 1, referenceDay) ) ;
28. // Navigate every day in this semester
29. int daySeqOfYear, daySeqOfWeek, weekSeqOfSemester ;
30. int currentYearLength;
31. currentYearLength = isLeapYear( currentYear );
32. for ( daySeqOfYear = sStartSeqOfYear, daySeqOfWeek = 0, weekSeqOfSemester = 1;
33. daySeqOfYear <= sEndSeqOfYear;
34. daySeqOfYear ++, daySeqOfWeek ++, daySeqOfWeek %= 7 )
35. {
36. // Before Monday, print the week sequence
37. if ( daySeqOfWeek == 0 )
38. {
39. printf("[%02d]", weekSeqOfSemester );
40. }
41. // Print the month and day for the current day
42. if ( daySeqOfYear <= currentYearLength )
43. {
44. printOneDay( currentYear, daySeqOfYear ) ;
45. }
46. else
47. {
48. printOneDay( currentYear + 1, daySeqOfYear ) ;
49. }
50. // After Sunday, print a new line
51. if ( daySeqOfWeek == 6 )
52. {
53. printf ("\n");
54. weekSeqOfSemester ++;
55. }
56. }
57. return 0;
58. }
59. int isLeapYear( int year )
60. {
61. if ( year == 2021 )
62. {
63. return 365;
64. }
65. }
66. int getMonthLength( int year, int month)
67. {
68. if (( year % 4 ) != 0 )
69. {
70. if ( month == 1 )
71. {
72. return 0;
73. }
74. if ( month == 2 )
75. {
76. return 31;
77. }
78. if ( month == 3 )
79. {
80. return ( 31 + 28 );
81. }
82. if ( month == 4 )
83. {
84. return ( 31 + 28 + 31 );
85. }
86. if ( month == 5 )
87. {
88. return ( 31 + 28 + 31 + 30 );
89. }
90. if ( month == 6 )
91. {
92. return ( 31 + 28 + 31 + 30 + 31 );
93. }
94. if ( month == 7 )
95. {
96. return ( 31 + 28 + 31 + 30 + 31 + 30 );
97. }
98. if ( month == 8 )
99. {
100. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 );
101. }
102. if ( month == 9 )
103. {
104. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 );
105. }
106. if ( month == 10 )
107. {
108. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 );
109. }
110. if ( month == 11 )
111. {
112. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 );
113. }
114. if ( month == 12 )
115. {
116. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 );
117. }
118. }
119. else
120. {
121. if ( month == 1 )
122. {
123. return 0;
124. }
125. if ( month == 2 )
126. {
127. return 31;
128. }
129. if ( month == 3 )
130. {
131. return ( 31 + 29 );
132. }
133. if ( month == 4 )
134. {
135. return ( 31 + 29 + 31 );
136. }
137. if ( month == 5 )
138. {
139. return ( 31 + 29 + 31 + 30 );
140. }
141. if ( month == 6 )
142. {
143. return ( 31 + 29 + 31 + 30 + 31 );
144. }
145. if ( month == 7 )
146. {
147. return ( 31 + 29 + 31 + 30 + 31 + 30 );
148. }
149. if ( month == 8 )
150. {
151. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 );
152. }
153. if ( month == 9 )
154. {
155. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 );
156. }
157. if ( month == 10 )
158. {
159. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 );
160. }
161. if ( month == 11 )
162. {
163. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 );
164. }
165. if ( month == 12 )
166. {
167. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 );
168. }
169. }
170. }
171. int getDaySeq( int year, int month, int day)
172. {
173. if ( year == 2021 )
174. {
175. return ( getMonthLength( year, month ) + day );
176. }
177. if ( year == 2022 )
178. {
179. return ( getMonthLength( year, month ) + day + isLeapYear( year - 1 ) );
180. }
181. }
182. int getDaySeqOfWeek( int year, int daySeqOfYear)
183. {
184. if ( ( daySeqOfYear + 4 ) % 7 != 0 )
185. {
186. return (( daySeqOfYear + 4 )% 7 );
187. }
188. else
189. {
190. return 7;
191. }
192. }
193. int getNextMonday( int year, int daySeqOfYear)
194. {
195. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 1 )
196. {
197. return daySeqOfYear;
198. }
199. else
200. {
201. while ( getDaySeqOfWeek( year, daySeqOfYear ) != 1 )
202. {
203. daySeqOfYear ++;
204. }
205. return daySeqOfYear;
206. }
207. }
208. int getThisMonday( int year, int daySeqOfYear)
209. {
210. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 1 )
211. {
212. return daySeqOfYear;
213. }
214. else
215. {
216. while ( getDaySeqOfWeek( year, daySeqOfYear ) != 1 )
217. {
218. daySeqOfYear --;
219. }
220. return daySeqOfYear;
221. }
222. }
223. int getThisSunday( int year, int daySeqOfYear)
224. {
225. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 7 )
226. {
227. return daySeqOfYear;
228. }
229. else
230. {
231. while ( getDaySeqOfWeek( year, daySeqOfYear ) != 7 )
232. {
233. daySeqOfYear ++;
234. }
235. return daySeqOfYear;
236. }
237. }
238. int getMonth( int year, int daySeqOfYear)
239. {
240. int data;
241. if ( year == 2021 )
242. {
243. data = daySeqOfYear;
244. }
245. if ( year == 2022 )
246. {
247. data = daySeqOfYear - isLeapYear( year - 1 );
248. }
249. int month;
250. int day;
251. if ( data <= 31 )
252. {
253. month = 1;
254. day = data;
255. return month;
256. }
257. if ( data <= 31 + 28 )
258. {
259. month = 2;
260. day = data - 31;
261. return month;
262. }
263. if ( data <= 31 + 28 + 31 )
264. {
265. month = 3;
266. day = data - 31 - 28;
267. return month;
268. }
269. if ( data <= 31 + 28 + 31 + 30 )
270. {
271. month = 4;
272. day = data - 31 - 28 - 31;
273. return month;
274. }
275. if ( data <= 31 + 28 + 31 + 30 + 31)
276. {
277. month = 5;
278. day = data - 31 - 28 - 31 -30;
279. return month;
280. }
281. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
282. {
283. month = 6;
284. day = data - 31 - 28 - 31 - 30 - 31;
285. return month;
286. }
287. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
288. {
289. month = 7;
290. day = data - 31 - 28 - 31 - 30 - 31 - 30;
291. return month;
292. }
293. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
294. {
295. month = 8;
296. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
297. return month;
298. }
299. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
300. {
301. month = 9;
302. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
303. return month;
304. }
305. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
306. {
307. month = 10;
308. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
309. return month;
310. }
311. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30)
312. {
313. month = 11;
314. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
315. return month;
316. }
317. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
318. {
319. month = 12;
320. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
321. return month;
322. }
323. }
324. int getDay( int year, int daySeqOfYear)
325. {
326. int data;
327. if ( year == 2021 )
328. {
329. data = daySeqOfYear;
330. }
331. if ( year == 2022 )
332. {
333. data = daySeqOfYear - isLeapYear( year - 1 );
334. }
335. int month;
336. int day;
337. if ( data <= 31 )
338. {
339. month = 1;
340. day = data;
341. return day;
342. }
343. if ( data <= 31 + 28 )
344. {
345. month = 2;
346. day = data - 31;
347. return day;
348. }
349. if ( data <= 31 + 28 + 31 )
350. {
351. month = 3;
352. day = data - 31 - 28;
353. return day;
354. }
355. if ( data <= 31 + 28 + 31 + 30 )
356. {
357. month = 4;
358. day = data - 31 - 28 - 31;
359. return day;
360. }
361. if ( data <= 31 + 28 + 31 + 30 + 31)
362. {
363. month = 5;
364. day = data - 31 - 28 - 31 -30;
365. return day;
366. }
367. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
368. {
369. month = 6;
370. day = data - 31 - 28 - 31 - 30 - 31;
371. return day;
372. }
373. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
374. {
375. month = 7;
376. day = data - 31 - 28 - 31 - 30 - 31 - 30;
377. return day;
378. }
379. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
380. {
381. month = 8;
382. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
383. return day;
384. }
385. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
386. {
387. month = 9;
388. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
389. return day;
390. }
391. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
392. {
393. month = 10;
394. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
395. return day;
396. }
397. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30)
398. {
399. month = 11;
400. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
401. return day;
402. }
403. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
404. {
405. month = 12;
406. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
407. return day;
408. }
409. }
410. void printOneDay( int year, int daySeqOfYear)
411. {
412. if ( getDay( year, daySeqOfYear) == 1 )
413. {
414. if ( getDaySeqOfWeek( year, daySeqOfYear) == 6 || getDaySeqOfWeek( year, daySeqOfYear) == 7 )
415. {
416. printf (" %02d.%02d!", getMonth( year, daySeqOfYear), getDay( year, daySeqOfYear));
417. }
418. else
419. {
420. printf (" %02d.%02d", getMonth( year, daySeqOfYear ), getDay( year, daySeqOfYear ));
421. }
422. }
423. else
424. {
425. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 6 || getDaySeqOfWeek( year, daySeqOfYear ) == 7 )
426. {
427. printf ("%9d!", getDay( year, daySeqOfYear ));
428. }
429. else
430. {
431. printf ("%10d", getDay( year, daySeqOfYear ));
432. }
433. }
434. }

（七） 2018到2021春秋校历

1. #include <stdio.h>
2. #include <stdlib.h>
3. int isLeapYear( int year );
4. // Functionsabout monthand day
5. int getMonthLength( int year, int month);
6. int getDaySeq( int year, int month, int day);
7. // Functions for propertiesofoneday
8. int getMonth( int year, int daySeqOfYear);
9. int getDay( int year, int daySeqOfYear);
10. int getDaySeqOfWeek( int year, int daySeqOfYear);
11. // Functions for daymovement calculation
12. int getNextMonday( int year, int daySeqOfYear);
13. int getThisSunday( int year, int daySeqOfYear);
14. int getThisMonday( int year, int daySeqOfYear);
15. void printOneDay( int year, int daySeqOfYear);
16. void printOneWeek( int year, int daySeqOfWeek, int weekSeqOfSemester,int daySeqOfYear);
17. // Calculate the first day and last day in the spring semester
18. int main(void)
19. {
20. int currentYear ;
21. printf ("请输入2018到2021年的一年\n");
22. scanf ("%d", &currentYear );
23. printf ("%d-%d年秋季校历\n", currentYear, currentYear + 1);
24. printf ("%4s%10s%10s%10s%10s%10s%10s%10s\n",
25. "#w:","Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
26. int sStartSeqOfYear1 = getThisMonday( currentYear, getDaySeq( currentYear, 9, 4) );
27. int sEndSeqOfYear1 = getThisSunday( currentYear + 1, getDaySeq( currentYear + 1, 1, 13) ) ;
28. // Navigate every day in this semester
29. int daySeqOfYear1, daySeqOfWeek1, weekSeqOfSemester1 ;
30. for ( daySeqOfYear1 = sStartSeqOfYear1, daySeqOfWeek1 = 0, weekSeqOfSemester1 = 1;
31. daySeqOfYear1 <= sEndSeqOfYear1;
32. daySeqOfYear1 ++, daySeqOfWeek1 ++, daySeqOfWeek1 %= 7 )
33. {
34. // Before Monday, print the week sequence
35. printOneWeek( currentYear, daySeqOfWeek1, weekSeqOfSemester1, daySeqOfYear1);
36. }
37. printf ("%d-%d年春季校历\n", currentYear, currentYear + 1);
38. printf ("%4s%10s%10s%10s%10s%10s%10s%10s\n",
39. "#w:","Mon.","Tues.","Wend.","Thur.","Fri", "Stat.","Sun.");
40. int sStartSeqOfYear2 = getThisMonday( currentYear + 1, getDaySeq( currentYear + 1, 2, 22) );
41. int sEndSeqOfYear2 = getThisSunday( currentYear + 1, getDaySeq( currentYear + 1, 7, 1) ) ;
42. // Navigate every day in this semester
43. int daySeqOfYear2, daySeqOfWeek2, weekSeqOfSemester2 ;
44. for ( daySeqOfYear2 = sStartSeqOfYear2, daySeqOfWeek2 = 0, weekSeqOfSemester2 = 1;
45. daySeqOfYear2 <= sEndSeqOfYear2;
46. daySeqOfYear2 ++, daySeqOfWeek2 ++, daySeqOfWeek2 %= 7 )
47. {
48. // Before Monday, print the week sequence
49. printOneWeek( currentYear, daySeqOfWeek2, weekSeqOfSemester2, daySeqOfYear2);
50. }
51. return 0;
52. }
53. int isLeapYear( int year )
54. {
55. if ( year == 2018 )
56. {
57. return 365;
58. }
59. if ( year == 2019 )
60. {
61. return 365 \* 2;
62. }
63. if ( year == 2020 )
64. {
65. return 365 \* 2 + 366;
66. }
67. if ( year == 2021 )
68. {
69. return 365 \* 3 + 366;
70. }
71. }
72. int getMonthLength( int year, int month)
73. {
74. if (( year % 4 ) != 0 )
75. {
76. if ( month == 1 )
77. {
78. return 0;
79. }
80. if ( month == 2 )
81. {
82. return 31;
83. }
84. if ( month == 3 )
85. {
86. return ( 31 + 28 );
87. }
88. if ( month == 4 )
89. {
90. return ( 31 + 28 + 31 );
91. }
92. if ( month == 5 )
93. {
94. return ( 31 + 28 + 31 + 30 );
95. }
96. if ( month == 6 )
97. {
98. return ( 31 + 28 + 31 + 30 + 31 );
99. }
100. if ( month == 7 )
101. {
102. return ( 31 + 28 + 31 + 30 + 31 + 30 );
103. }
104. if ( month == 8 )
105. {
106. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 );
107. }
108. if ( month == 9 )
109. {
110. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 );
111. }
112. if ( month == 10 )
113. {
114. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 );
115. }
116. if ( month == 11 )
117. {
118. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 );
119. }
120. if ( month == 12 )
121. {
122. return ( 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 );
123. }
124. }
125. else
126. {
127. if ( month == 1 )
128. {
129. return 0;
130. }
131. if ( month == 2 )
132. {
133. return 31;
134. }
135. if ( month == 3 )
136. {
137. return ( 31 + 29 );
138. }
139. if ( month == 4 )
140. {
141. return ( 31 + 29 + 31 );
142. }
143. if ( month == 5 )
144. {
145. return ( 31 + 29 + 31 + 30 );
146. }
147. if ( month == 6 )
148. {
149. return ( 31 + 29 + 31 + 30 + 31 );
150. }
151. if ( month == 7 )
152. {
153. return ( 31 + 29 + 31 + 30 + 31 + 30 );
154. }
155. if ( month == 8 )
156. {
157. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 );
158. }
159. if ( month == 9 )
160. {
161. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 );
162. }
163. if ( month == 10 )
164. {
165. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 );
166. }
167. if ( month == 11 )
168. {
169. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 );
170. }
171. if ( month == 12 )
172. {
173. return ( 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 );
174. }
175. }
176. }
177. int getDaySeq( int year, int month, int day)
178. {
179. if ( year == 2018 )
180. {
181. return ( getMonthLength( year, month ) + day );
182. }
183. if ( year >= 2018 )
184. {
185. return ( getMonthLength( year, month ) + day + isLeapYear( year - 1 ) );
186. }
187. }
188. int getDaySeqOfWeek( int year, int daySeqOfYear)
189. {
190. if ( ( daySeqOfYear ) % 7 != 0 )
191. {
192. return (( daySeqOfYear )% 7 );
193. }
194. else
195. {
196. return 7;
197. }
198. }
199. int getNextMonday( int year, int daySeqOfYear)
200. {
201. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 1 )
202. {
203. return daySeqOfYear;
204. }
205. else
206. {
207. while ( getDaySeqOfWeek( year, daySeqOfYear ) != 1 )
208. {
209. daySeqOfYear ++;
210. }
211. return daySeqOfYear;
212. }
213. }
214. int getThisMonday( int year, int daySeqOfYear)
215. {
216. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 1 )
217. {
218. return daySeqOfYear;
219. }
220. else
221. {
222. while ( getDaySeqOfWeek( year, daySeqOfYear ) != 1 )
223. {
224. daySeqOfYear --;
225. }
226. return daySeqOfYear;
227. }
228. }
229. int getThisSunday( int year, int daySeqOfYear)
230. {
231. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 7 )
232. {
233. return daySeqOfYear;
234. }
235. else
236. {
237. while ( getDaySeqOfWeek( year, daySeqOfYear ) != 7 )
238. {
239. daySeqOfYear ++;
240. }
241. return daySeqOfYear;
242. }
243. }
244. int getMonth( int year, int daySeqOfYear)
245. {
246. int data;
247. if ( year == 2018 )
248. {
249. data = daySeqOfYear;
250. }
251. else
252. {
253. data = daySeqOfYear - isLeapYear( year - 1 );
254. }
255. int month;
256. int day;
257. if ( year != 2020 )
258. {
259. if ( data <= 31 )
260. {
261. month = 1;
262. day = data;
263. return month;
264. }
265. if ( data <= 31 + 28 )
266. {
267. month = 2;
268. day = data - 31;
269. return month;
270. }
271. if ( data <= 31 + 28 + 31 )
272. {
273. month = 3;
274. day = data - 31 - 28;
275. return month;
276. }
277. if ( data <= 31 + 28 + 31 + 30 )
278. {
279. month = 4;
280. day = data - 31 - 28 - 31;
281. return month;
282. }
283. if ( data <= 31 + 28 + 31 + 30 + 31)
284. {
285. month = 5;
286. day = data - 31 - 28 - 31 -30;
287. return month;
288. }
289. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
290. {
291. month = 6;
292. day = data - 31 - 28 - 31 - 30 - 31;
293. return month;
294. }
295. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
296. {
297. month = 7;
298. day = data - 31 - 28 - 31 - 30 - 31 - 30;
299. return month;
300. }
301. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
302. {
303. month = 8;
304. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
305. return month;
306. }
307. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
308. {
309. month = 9;
310. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
311. return month;
312. }
313. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
314. {
315. month = 10;
316. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
317. return month;
318. }
319. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30)
320. {
321. month = 11;
322. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
323. return month;
324. }
325. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
326. {
327. month = 12;
328. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
329. return month;
330. }
331. }
332. else
333. {
334. if ( data <= 31 )
335. {
336. month = 1;
337. day = data;
338. return month;
339. }
340. if ( data <= 31 + 29 )
341. {
342. month = 2;
343. day = data - 31;
344. return month;
345. }
346. if ( data <= 31 + 29 + 31 )
347. {
348. month = 3;
349. day = data - 31 - 29;
350. return month;
351. }
352. if ( data <= 31 + 29 + 31 + 30 )
353. {
354. month = 4;
355. day = data - 31 - 29 - 31;
356. return month;
357. }
358. if ( data <= 31 + 29 + 31 + 30 + 31)
359. {
360. month = 5;
361. day = data - 31 - 29 - 31 -30;
362. return month;
363. }
364. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 )
365. {
366. month = 6;
367. day = data - 31 - 29 - 31 - 30 - 31;
368. return month;
369. }
370. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 )
371. {
372. month = 7;
373. day = data - 31 - 29 - 31 - 30 - 31 - 30;
374. return month;
375. }
376. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 )
377. {
378. month = 8;
379. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31;
380. return month;
381. }
382. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
383. {
384. month = 9;
385. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31;
386. return month;
387. }
388. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
389. {
390. month = 10;
391. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
392. return month;
393. }
394. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30)
395. {
396. month = 11;
397. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
398. return month;
399. }
400. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
401. {
402. month = 12;
403. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
404. return month;
405. }
406. }
407. }
408. int getDay( int year, int daySeqOfYear)
409. {
410. int data;
411. if ( year == 2018 )
412. {
413. data = daySeqOfYear;
414. }
415. else
416. {
417. data = daySeqOfYear - isLeapYear( year - 1 );
418. }
419. int month;
420. int day;
421. if ( year != 2020 )
422. {
423. if ( data <= 31 )
424. {
425. month = 1;
426. day = data;
427. return day;
428. }
429. if ( data <= 31 + 28 )
430. {
431. month = 2;
432. day = data - 31;
433. return day;
434. }
435. if ( data <= 31 + 28 + 31 )
436. {
437. month = 3;
438. day = data - 31 - 28;
439. return day;
440. }
441. if ( data <= 31 + 28 + 31 + 30 )
442. {
443. month = 4;
444. day = data - 31 - 28 - 31;
445. return day;
446. }
447. if ( data <= 31 + 28 + 31 + 30 + 31)
448. {
449. month = 5;
450. day = data - 31 - 28 - 31 -30;
451. return day;
452. }
453. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 )
454. {
455. month = 6;
456. day = data - 31 - 28 - 31 - 30 - 31;
457. return day;
458. }
459. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 )
460. {
461. month = 7;
462. day = data - 31 - 28 - 31 - 30 - 31 - 30;
463. return day;
464. }
465. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 )
466. {
467. month = 8;
468. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31;
469. return day;
470. }
471. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
472. {
473. month = 9;
474. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31;
475. return day;
476. }
477. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
478. {
479. month = 10;
480. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
481. return day;
482. }
483. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30)
484. {
485. month = 11;
486. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
487. return day;
488. }
489. if ( data <= 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
490. {
491. month = 12;
492. day = data - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
493. return day;
494. }
495. }
496. else
497. {
498. if ( data <= 31 )
499. {
500. month = 1;
501. day = data;
502. return day;
503. }
504. if ( data <= 31 + 29 )
505. {
506. month = 2;
507. day = data - 31;
508. return day;
509. }
510. if ( data <= 31 + 29 + 31 )
511. {
512. month = 3;
513. day = data - 31 - 29;
514. return day;
515. }
516. if ( data <= 31 + 29 + 31 + 30 )
517. {
518. month = 4;
519. day = data - 31 - 29 - 31;
520. return day;
521. }
522. if ( data <= 31 + 29 + 31 + 30 + 31)
523. {
524. month = 5;
525. day = data - 31 - 29 - 31 -30;
526. return day;
527. }
528. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 )
529. {
530. month = 6;
531. day = data - 31 - 29 - 31 - 30 - 31;
532. return day;
533. }
534. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 )
535. {
536. month = 7;
537. day = data - 31 - 29 - 31 - 30 - 31 - 30;
538. return day;
539. }
540. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 )
541. {
542. month = 8;
543. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31;
544. return day;
545. }
546. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 )
547. {
548. month = 9;
549. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31;
550. return day;
551. }
552. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 )
553. {
554. month = 10;
555. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31 - 30;
556. return day;
557. }
558. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30)
559. {
560. month = 11;
561. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31;
562. return day;
563. }
564. if ( data <= 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + 31 )
565. {
566. month = 12;
567. day = data - 31 - 29 - 31 - 30 - 31 - 30 - 31 - 31 - 30 - 31 - 30;
568. return day;
569. }
570. }
571. }
572. void printOneDay( int year, int daySeqOfYear)
573. {
574. if ( (getMonth( year, daySeqOfYear) == 1 && getDay( year, daySeqOfYear) == 1 ) ||
575. daySeqOfYear == getThisMonday ( year, getDaySeq( year, 9, 4)) || daySeqOfYear == getThisSunday ( year, getDaySeq( year, 1,13)) ||
576. daySeqOfYear == getThisMonday ( year, getDaySeq( year, 2,22)) || daySeqOfYear == getThisSunday ( year, getDaySeq( year, 7, 1)))
577. {
578. if ( getDaySeqOfWeek( year, daySeqOfYear) == 6 || getDaySeqOfWeek( year, daySeqOfYear) == 7 )
579. {
580. printf (" %02d.%02d.%02d!", year % 100, getMonth( year, daySeqOfYear), getDay( year, daySeqOfYear));
581. }
582. else
583. {
584. printf (" %02d.%02d.%02d", year % 100, getMonth( year, daySeqOfYear ), getDay( year, daySeqOfYear ));
585. }
586. }
587. else
588. {
589. if ( getDay( year, daySeqOfYear) == 1 )
590. {
591. if ( getDaySeqOfWeek( year, daySeqOfYear) == 6 || getDaySeqOfWeek( year, daySeqOfYear) == 7 )
592. {
593. printf (" %02d.%02d!", getMonth( year, daySeqOfYear), getDay( year, daySeqOfYear));
594. }
595. else
596. {
597. printf (" %02d.%02d", getMonth( year, daySeqOfYear ), getDay( year, daySeqOfYear ));
598. }
599. }
600. else
601. {
602. if ( getDaySeqOfWeek( year, daySeqOfYear ) == 6 || getDaySeqOfWeek( year, daySeqOfYear ) == 7 )
603. {
604. printf ("%9d!", getDay( year, daySeqOfYear ));
605. }
606. else
607. {
608. printf ("%10d", getDay( year, daySeqOfYear ));
609. }
610. }
611. }
612. }
613. void printOneWeek( int currentYear, int daySeqOfWeek, int weekSeqOfSemester, int daySeqOfYear)
614. {
615. if ( daySeqOfWeek == 0 )
616. {
617. printf("[%02d]", weekSeqOfSemester );
618. }
619. // Print the month and day for the current day
620. if ( daySeqOfYear <= isLeapYear( currentYear ) )
621. {
622. printOneDay( currentYear, daySeqOfYear ) ;
623. }
624. else
625. {
626. printOneDay( currentYear + 1, daySeqOfYear ) ;
627. }
628. // After Sunday, print a new line
629. if ( daySeqOfWeek == 6 )
630. {
631. printf ("\n");
632. weekSeqOfSemester ++;
633. }
634. }

（八） 2000到2025任给一年输出后五年秋季校历第一周，并比较开学日期早晚

1. #include <stdio.h>
2. #include <stdlib.h>
3. #define YEAR\_NUM 5
4. #define YEAR\_MIN 2000
5. #define YEAR\_MAX 2030
7. **int** getThisMonday( **int** year, **int** daySeqOfYear, **int** inputYear);
8. **int** isLeapYear( **int** year, **int** inputYear );
9. **int** getDaySeq( **int** year, **int** month, **int** day, **int** inputYear);
10. **int** getDaySeqOfWeek( **int** year, **int** daySeqOfYear, **int** inputYear);
11. **int** getMonthLength( **int** year, **int** month);
12. **void** printOneWeek( **int** year, **int** dlay, **int** inputYear );
13. **int** getMonth ( **int** year, **int** dlay, **int** inputYear );
14. **int** getDay ( **int** year, **int** dlay, **int** inputYear );

17. **const** **int** Month\_NORMAL\_YEAR [12] = {31,28,31,30,31,30,31,31,30,31,30,31};
18. **const** **int** Month\_LEAP\_YEAR [12] = {31,29,31,30,31,30,31,31,30,31,30,31};
20. **int** main (**void**)
21. {
22. // Declare variables
23. **int** inputYear,Years[YEAR\_NUM];
24. // Display the program information
25. printf("Spring Calendar Comparision in %d years, please input the first year (%d-%d):\n",
26. YEAR\_NUM, YEAR\_MIN, YEAR\_MAX - YEAR\_NUM );
27. scanf("%d", &inputYear );
28. **if** ( inputYear < YEAR\_MIN || inputYear + YEAR\_NUM > YEAR\_MAX)
29. {
30. printf ("Sorry, the input year is not supported.\n");
31. **return** 1;
32. }
33. **int** j = 0;
34. **for** ( ;j < 5; j ++ )
35. {
36. Years[j] = inputYear + j;
37. }

40. **int** i;
41. **int** startDays[YEAR\_NUM] = {0};
42. **for** ( i = 0; i < YEAR\_NUM; i++ )
43. {
44. printf("\n%s%s%d\n"," ","First week in spring calendar of Year ", Years[i]);
45. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.","Tues.","wed.","Thur.","Fri.","sat.","sun.");
46. printOneWeek( Years[i], getThisMonday( Years[i],getDaySeq( Years[i], 9, 4, inputYear), inputYear), inputYear);
47. **if** ( i == 0 )
48. {
49. startDays[i] = getThisMonday( Years[i],getDaySeq( Years[i], 9, 4, inputYear), inputYear);
50. }
51. **else**
52. {
53. startDays[i] = getThisMonday( Years[i],getDaySeq( Years[i], 9, 4, inputYear), inputYear ) - isLeapYear( Years[i - 1], inputYear);
54. }
55. }
56. **int** min = 0;
57. **for** ( i = 1; i < YEAR\_NUM; i ++ )
58. {
59. min = ( startDays[i] < startDays[min]) ? i : min;
60. }
61. printf ("\n%s%s%d\n"," ","Earliest spring semester is in Year ", Years[min]);
62. }
64. **int** getThisMonday( **int** year, **int** daySeqOfYear, **int** inputYear)
65. {
66. **if** ( getDaySeqOfWeek( year, daySeqOfYear, inputYear ) == 1 )
67. {
68. **return** daySeqOfYear;
69. }
70. **else**
71. {
72. **while** ( getDaySeqOfWeek( year, daySeqOfYear, inputYear ) != 1 )
73. {
74. daySeqOfYear --;
75. }
76. **return** daySeqOfYear;
77. }
78. }
80. **int** isLeapYear( **int** year, **int** inputYear )
81. {
82. **int** length = 0;
83. **for** ( ; year >= inputYear; year -- )
84. {
85. **if** ( year % 4 != 0 )
86. {
87. length = length + 365;
88. }
89. **else**
90. {
91. length = length + 366;
92. }
93. }
94. **return** length;
95. }
97. **int** getDaySeq( **int** year, **int** month, **int** day, **int** inputYear)
98. {
99. **if** ( year == inputYear )
100. {
101. **return** ( getMonthLength( year, month ) + day );
102. }
103. **if** ( year > inputYear )
104. {
105. **return** ( getMonthLength( year, month ) + day + isLeapYear( year - 1, inputYear ) );
106. }
108. }
110. **int** getDaySeqOfWeek( **int** year, **int** daySeqOfYear, **int** inputYear)
111. {
112. **int** first;//这个是求输入年份的元旦前一天的周几
113. first = ( isLeapYear( 2019, inputYear - 1 ) + 1 ) % 7;
114. **if** ( ( daySeqOfYear + first ) % 7 != 0 )
115. {
116. **return** (( daySeqOfYear + first )% 7 );
117. }
118. **else**
119. {
120. **return** 7;
121. }
122. }
123. **int** getMonthLength( **int** year, **int** month)
124. {
125. **int** add = 0;
126. **if** ( year % 4 != 0 )
127. {
128. **for** ( ; month >= 2; month -- )
129. {
130. add = add + Month\_NORMAL\_YEAR [month - 2];
131. }
132. }
133. **else**
134. {
135. **for** ( ; month >= 2; month -- )
136. {
137. add = add + Month\_LEAP\_YEAR [month - 2];
138. }
139. }
140. **return** add;
141. }
143. **void** printOneWeek( **int** year, **int** dlay, **int** inputYear )
144. {
145. printf ("[01]");
146. printf ("  %02d.%02d.%02d", year % 100, getMonth (year,dlay,inputYear), getDay (year,dlay,inputYear));
147. **int** integer = 1;
148. **for** ( ;integer < 7; integer ++ )
149. {
150. **if** ( getDay (year,dlay + integer,inputYear) == 1)
151. printf ("     09.01");
152. **else**
153. {
154. **if** ( integer == 5 || integer == 6)
155. {
156. printf ("       %2d!",getDay (year,dlay + integer,inputYear));
157. }
158. **else**
159. {
160. printf ("%10d",getDay (year,dlay + integer,inputYear));
161. }
162. }
163. }
164. }
166. **int** getMonth ( **int** year, **int** dlay, **int** inputYear )
167. {
168. **int** common;
169. **if** ( year == inputYear )
170. {
171. common = dlay;
172. }
173. **if** ( year > inputYear )
174. {
175. common = dlay - isLeapYear( year - 1, inputYear ) ;
176. }
177. **int** month = 0;
178. **if** ( year % 4 != 0 )
179. {
180. **for** ( ; common > 0; month ++ )
181. {
182. common = common - Month\_NORMAL\_YEAR [month];
183. }
184. **return** month;
185. }
186. **else**
187. {
188. **for** ( ; common > 0; month ++ )
189. {
190. common = common - Month\_LEAP\_YEAR [month];
191. }
192. **return** month;
193. }
194. }
196. **int** getDay ( **int** year, **int** dlay, **int** inputYear )
197. {
198. **int** common;
199. **if** ( year == inputYear )
200. {
201. common = dlay;
202. }
203. **if** ( year > inputYear )
204. {
205. common = dlay - isLeapYear( year - 1, inputYear ) ;
206. }
207. **int** month = 0;
208. **if** ( year % 4 != 0 )
209. {
210. **for** ( ; common > 0; month ++ )
211. {
212. common = common - Month\_NORMAL\_YEAR [month];
213. }
214. **return** (common + Month\_NORMAL\_YEAR [month - 1]);
215. }
216. **else**
217. {
218. **for** ( ; common > 0; month ++ )
219. {
220. common = common - Month\_LEAP\_YEAR [month];
221. }
222. **return** (common + Month\_LEAP\_YEAR [month - 1]);
223. }
224. }

（九） 加入数组，输出从2000到2007生日那周的周历，并算出有多少在周末过

1. #include <stdio.h>
2. #include <stdlib.h>
3. #define YEAR\_MIN 2000
4. #define YEAR\_MAX 2007
5. int getThisMonday( int year, int daySeqOfYear, int inputYear);
6. int isLeapYear( int year, int inputYear );
7. int getDaySeq( int year, int month, int day, int inputYear);
8. int getDaySeqOfWeek( int year, int daySeqOfYear, int inputYear);
9. int getMonthLength( int year, int month);
10. void printOneWeek( int year, int dlay, int inputYear );
11. int getMonth ( int year, int dlay, int inputYear );
12. int getDay ( int year, int dlay, int inputYear );
13. void initialDays(int Years[], int Days[][366][4], int inputYear );
14. const int Month\_NORMAL\_YEAR [12] = {31,28,31,30,31,30,31,31,30,31,30,31};
15. const int Month\_LEAP\_YEAR [12] = {31,29,31,30,31,30,31,31,30,31,30,31};
16. int main (void)
17. {
18. // Declare variables
19. int inputYear = YEAR\_MIN;
20. int Years[YEAR\_MAX - YEAR\_MIN + 1];
21. int inputMonth, inputDay;
22. // Display the program information
23. printf("Find birthday in year(%d-%d), please input the month and day.\n",
24. YEAR\_MIN, YEAR\_MAX );
25. scanf("%d %d", &inputMonth, &inputDay );
26. if ( inputMonth < 1 ||inputMonth > 12 || inputDay < 1 || inputDay > 31)
27. {
28. printf("Sorry, the input month and day are invalid.\n");
29. return 1;
30. }
31. int j = 0;
32. for ( ;j < 8; j ++ )
33. {
34. Years[j] = inputYear + j;
35. }
36. int Days[YEAR\_MAX - YEAR\_MIN + 1][366][4] = {0};
37. initialDays(Years, Days, inputYear );
38. int i;
39. int startDays[YEAR\_MAX - YEAR\_MIN + 1] = {0};
40. for ( i = 0; i < YEAR\_MAX - YEAR\_MIN + 1; i++ )
41. {
42. printf("\n%s%s%d\n"," ","Birthday in year ", Years[i]);
43. printf(" #M:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.","Tues.","wed.","Thur.","Fri.","sat.","sun.");
44. printOneWeek( Years[i] , getThisMonday( Years[i],getDaySeq( Years[i], inputMonth, inputDay, inputYear), inputYear), inputYear);
45. if ( i == 0 )
46. {
47. startDays[i] = getThisMonday( Years[i],getDaySeq( Years[i], inputMonth, inputDay, inputYear), inputYear);
48. }
49. else
50. {
51. startDays[i] = getThisMonday( Years[i],getDaySeq( Years[i], inputMonth, inputDay, inputYear), inputYear ) - isLeapYear( Years[i - 1], inputYear);
52. }
53. }
54. int circle = 0;
55. for ( i = 0; i < YEAR\_MAX - YEAR\_MIN + 1; i ++ )
56. {
57. if ( getDaySeqOfWeek( Years[i], getDaySeq( Years[i], inputMonth, inputDay, inputYear), inputYear) == 6 ||
58. getDaySeqOfWeek( Years[i], getDaySeq( Years[i], inputMonth, inputDay, inputYear), inputYear) == 7 )
59. {
60. circle ++;
61. }
62. }
63. printf ("\nWe find %d,%d are on weekends.\n", YEAR\_MAX - YEAR\_MIN + 1,circle);
64. }
65. int getThisMonday( int year, int daySeqOfYear, int inputYear)
66. {
67. if ( getDaySeqOfWeek( year, daySeqOfYear, inputYear ) == 1 )
68. {
69. return daySeqOfYear;
70. }
71. else
72. {
73. while ( getDaySeqOfWeek( year, daySeqOfYear, inputYear ) != 1 )
74. {
75. daySeqOfYear --;
76. }
77. return daySeqOfYear;
78. }
79. }
80. int isLeapYear( int year, int inputYear )
81. {
82. int length = 0;
83. for ( ; year >= inputYear; year -- )
84. {
85. if ( year % 4 != 0 )
86. {
87. length = length + 365;
88. }
89. else
90. {
91. length = length + 366;
92. }
93. }
94. return length;
95. }
96. int getDaySeq( int year, int month, int day, int inputYear)
97. {
98. if ( year == inputYear )
99. {
100. return ( getMonthLength( year, month ) + day );
101. }
102. if ( year > inputYear )
103. {
104. return ( getMonthLength( year, month ) + day + isLeapYear( year - 1, inputYear ) );
105. }
106. }
107. int getDaySeqOfWeek( int year, int daySeqOfYear, int inputYear)
108. {
109. int first;//这个是求输入年份的元旦前一天的周几
110. first = ( isLeapYear( 2019, inputYear - 1 ) + 1 ) % 7;
111. if ( ( daySeqOfYear + first ) % 7 != 0 )
112. {
113. return (( daySeqOfYear + first )% 7 );
114. }
115. else
116. {
117. return 7;
118. }
119. }
120. int getMonthLength( int year, int month)
121. {
122. int add = 0;
123. if ( year % 4 != 0 )
124. {
125. for ( ; month >= 2; month -- )
126. {
127. add = add + Month\_NORMAL\_YEAR [month - 2];
128. }
129. }
130. else
131. {
132. for ( ; month >= 2; month -- )
133. {
134. add = add + Month\_LEAP\_YEAR [month - 2];
135. }
136. }
137. return add;
138. }
139. void printOneWeek( int year, int dlay, int inputYear )
140. {
141. printf ("[%02d]",getMonth ( year, dlay, inputYear));
142. int integer = 0;
143. for ( ;integer < 7; integer ++ )
144. {
145. if ( integer == 5 || integer == 6)
146. {
147. printf (" %2d!",getDay (year,dlay + integer,inputYear));
148. }
149. else
150. {
151. printf ("%10d",getDay (year,dlay + integer,inputYear));
152. }
153. }
154. }
155. int getMonth ( int year, int dlay, int inputYear )
156. {
157. int common;
158. if ( year == inputYear )
159. {
160. common = dlay;
161. }
162. if ( year > inputYear )
163. {
164. common = dlay - isLeapYear( year - 1, inputYear ) ;
165. }
166. int month = 0;
167. if ( year % 4 != 0 )
168. {
169. for ( ; common > 0; month ++ )
170. {
171. common = common - Month\_NORMAL\_YEAR [month];
172. }
173. return month;
174. }
175. else
176. {
177. for ( ; common > 0; month ++ )
178. {
179. common = common - Month\_LEAP\_YEAR [month];
180. }
181. return month;
182. }
183. }
184. int getDay ( int year, int dlay, int inputYear )
185. {
186. int common;
187. if ( year == inputYear )
188. {
189. common = dlay;
190. }
191. if ( year > inputYear )
192. {
193. common = dlay - isLeapYear( year - 1, inputYear ) ;
194. }
195. int month = 0;
196. if ( year % 4 != 0 )
197. {
198. for ( ; common > 0; month ++ )
199. {
200. common = common - Month\_NORMAL\_YEAR [month];
201. }
202. return (common + Month\_NORMAL\_YEAR [month - 1]);
203. }
204. else
205. {
206. for ( ; common > 0; month ++ )
207. {
208. common = common - Month\_LEAP\_YEAR [month];
209. }
210. return (common + Month\_LEAP\_YEAR [month - 1]);
211. }
212. }
213. void initialDays(int Years[], int Days[][366][4], int inputYear )
214. {
215. int year, month, day, yearLength, weekseq,seqofweek;
216. int i, j;
217. for(i = 0 ; i < YEAR\_MAX - YEAR\_MIN + 1; i ++ )
218. {
219. year = Years[i];
220. if ( year % 4 == 0 )
221. {
222. yearLength = 366;
223. }
224. else
225. {
226. yearLength = 365;
227. }
228. for ( j = 0 ; j < yearLength; j ++ )
229. {
230. Days[i][j][0] = getMonth( year, j + 1 + isLeapYear( year - 1, inputYear), inputYear);
231. Days[i][j][1] = getDay( year, j + 1 + isLeapYear( year - 1, inputYear), inputYear);
232. Days[i][j][2] = j + 1 + isLeapYear( year - 1, inputYear);
233. Days[i][j][3] = getDaySeqOfWeek( year, j + 1 + isLeapYear( year - 1, inputYear), inputYear);
234. }
235. }
236. }

（十） 寻找生日，结构体版

1. #include <stdio.h>
2. #include <stdlib.h>
3. #define YEAR\_MIN 2000
4. #define YEAR\_MAX 2007
5. typedef struct{
6. int year;
7. int daySeq;
8. int month;
9. int day;
10. int weekSeq;
11. int weekDay;
12. }Day;
13. int isLeapYear( int year, int inputYear );
14. int getDaySeq( int year, int month, int day, int inputYear);
15. int getDaySeqOfWeek( int year, int daySeqOfYear, int inputYear);
16. int getWeekSeqOfYear( int year, int daySeqOfYear, int inputYear);
17. int getMonthLength( int year, int month);
18. int getMonth ( int year, int dlay, int inputYear );
19. int getDay ( int year, int dlay, int inputYear );
20. Day setDay ( int year, int month, int day, int inputYear);
21. Day moveDay ( Day day1, int inputYear );
22. void printfDay ( Day birthday, Day before1, Day before2, Day setday);
23. const int Month\_NORMAL\_YEAR [12] = {31,28,31,30,31,30,31,31,30,31,30,31};
24. const int Month\_LEAP\_YEAR [12] = {31,29,31,30,31,30,31,31,30,31,30,31};
25. int main (void)
26. {
27. // Declare variables
28. int inputYear = YEAR\_MIN;
29. int Years[YEAR\_MAX - YEAR\_MIN + 1];
30. int inputMonth, inputDay;
31. // Display the program information
32. printf("Prepare birthday in year(%d-%d), please input the month and day.\n",
33. YEAR\_MIN, YEAR\_MAX );
34. scanf("%d %d", &inputMonth, &inputDay );
35. int j = 0;
36. for ( ;j < 8; j ++ )
37. {
38. Years[j] = inputYear + j;
39. }
40. int i;
41. int startDays[YEAR\_MAX - YEAR\_MIN + 1] = {0};
42. for ( i = 0; i < YEAR\_MAX - YEAR\_MIN + 1; i++ )
43. {
44. printf("\n%s%s%d\n"," ","Birthday in year ", Years[i]);
45. if ( inputMonth == 2 && inputDay == 29 && Years[i] % 4 != 0 )
46. {
47. printf ("Not found.\n");
48. }
49. else
50. {
51. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.","Tues.","wed.","Thur.","Fri.","sat.","sun.");
52. Day birthday = setDay ( Years[i], inputMonth, inputDay, inputYear);
53. Day before1 = moveDay ( birthday, inputYear);
54. Day before2 = moveDay ( before1, inputYear);
55. Day setday = moveDay ( before2, inputYear);
56. printfDay ( birthday, before1, before2, setday);
57. }
58. }
59. }
60. int isLeapYear( int year, int inputYear )
61. {
62. int length = 0;
63. for ( ; year >= inputYear; year -- )
64. {
65. if ( year % 4 != 0 )
66. {
67. length = length + 365;
68. }
69. else
70. {
71. length = length + 366;
72. }
73. }
74. return length;
75. }
76. int getDaySeq( int year, int month, int day, int inputYear)
77. {
78. if ( year == inputYear )
79. {
80. return ( getMonthLength( year, month ) + day );
81. }
82. if ( year > inputYear )
83. {
84. return ( getMonthLength( year, month ) + day + isLeapYear( year - 1, inputYear ) );
85. }
86. }
87. int getDaySeqOfWeek( int year, int daySeqOfYear, int inputYear)
88. {
89. int first;//这个是求输入年份的元旦前一天的周几
90. first = ( isLeapYear( 2019, inputYear - 1 ) + 1 ) % 7;
91. if ( ( daySeqOfYear + first ) % 7 != 0 )
92. {
93. return (( daySeqOfYear + first )% 7 );
94. }
95. else
96. {
97. return 7;
98. }
99. }
100. int getMonthLength( int year, int month)
101. {
102. int add = 0;
103. if ( year % 4 != 0 )
104. {
105. for ( ; month >= 2; month -- )
106. {
107. add = add + Month\_NORMAL\_YEAR [month - 2];
108. }
109. }
110. else
111. {
112. for ( ; month >= 2; month -- )
113. {
114. add = add + Month\_LEAP\_YEAR [month - 2];
115. }
116. }
117. return add;
118. }
119. int getMonth ( int year, int dlay, int inputYear )
120. {
121. int common;
122. if ( year == inputYear )
123. {
124. common = dlay;
125. }
126. if ( year > inputYear )
127. {
128. common = dlay - isLeapYear( year - 1, inputYear ) ;
129. }
130. int month = 0;
131. if ( year % 4 != 0 )
132. {
133. for ( ; common > 0; month ++ )
134. {
135. common = common - Month\_NORMAL\_YEAR [month];
136. }
137. return month;
138. }
139. else
140. {
141. for ( ; common > 0; month ++ )
142. {
143. common = common - Month\_LEAP\_YEAR [month];
144. }
145. return month;
146. }
147. }
148. int getDay ( int year, int dlay, int inputYear )
149. {
150. int common;
151. if ( year == inputYear )
152. {
153. common = dlay;
154. }
155. if ( year > inputYear )
156. {
157. common = dlay - isLeapYear( year - 1, inputYear ) ;
158. }
159. int month = 0;
160. if ( year % 4 != 0 )
161. {
162. for ( ; common > 0; month ++ )
163. {
164. common = common - Month\_NORMAL\_YEAR [month];
165. }
166. return (common + Month\_NORMAL\_YEAR [month - 1]);
167. }
168. else
169. {
170. for ( ; common > 0; month ++ )
171. {
172. common = common - Month\_LEAP\_YEAR [month];
173. }
174. return (common + Month\_LEAP\_YEAR [month - 1]);
175. }
176. }
177. int getWeekSeqOfYear( int year, int daySeqOfYear, int inputYear)
178. {
179. return ( daySeqOfYear + 5 - getDaySeqOfWeek( year, daySeqOfYear, inputYear)) / 7 + 1 - isLeapYear( year - 1, inputYear ) / 7;
180. }
181. Day setDay ( int year, int month, int day, int inputYear)
182. {
183. Day day1;
184. day1.year = year;
185. day1.daySeq = getDaySeq( year, month, day, inputYear);
186. day1.month = month;
187. day1.day = day;
188. day1.weekSeq = getWeekSeqOfYear( year, day1.daySeq, inputYear);
189. day1.weekDay = getDaySeqOfWeek( year, day1.daySeq, inputYear);
190. return day1;
191. }
192. Day moveDay ( Day day1, int inputYear )
193. {
194. Day day2;
195. if ( day1.month == 1 && day1.day == 1 )
196. {
197. day2.year = day1.year - 1;
198. }
199. else
200. {
201. day2.year = day1.year;
202. }
203. day2.daySeq = day1.daySeq - 1;
204. day2.month = getMonth ( day2.year, day2.daySeq, inputYear );
205. day2.day = getDay ( day2.year, day2.daySeq, inputYear );
206. day2.weekSeq = getWeekSeqOfYear( day2.year, day2.daySeq, inputYear);
207. day2.weekDay = getDaySeqOfWeek( day2.year, day2.daySeq, inputYear);
208. return day2;
209. }
210. void printfDay ( Day birthday, Day before1, Day before2, Day setDay)
211. {
212. if ( setDay.weekDay == 1 )
213. {
214. printf ("[%02d] %02d.%02d.%02d%10d%10d %02d.%02d\*\n", setDay.weekSeq, setDay.year % 100, setDay.month,
215. setDay.day, before2.day, before1.day, birthday.month, birthday.day);
216. }
217. if ( setDay.weekDay == 2 )
218. {
219. printf ("[%02d] %2d.%02d.%02d%10d%10d %02d.%02d\*\n", setDay.weekSeq, setDay.year % 100, setDay.month,
220. setDay.day, before2.day, before1.day, birthday.month, birthday.day);
221. }
222. if ( setDay.weekDay == 3 )
223. {
224. printf ("[%02d] %02d.%02d.%02d%10d%10d %02d.%02d\*\n", setDay.weekSeq, setDay.year % 100, setDay.month,
225. setDay.day, before2.day, before1.day, birthday.month, birthday.day);
226. }
227. if ( setDay.weekDay == 4 )
228. {
229. printf ("[%02d] %02d.%02d.%02d%10d%10d %02d.%02d\*\n",
230. setDay.weekSeq, setDay.year % 100, setDay.month,
231. setDay.day, before2.day, before1.day, birthday.month, birthday.day);
232. }
233. if ( setDay.weekDay == 5 )
234. {
235. printf ("[%02d] %02d.%02d.%02d%10d%10d\n[%02d] %02d.%02d\*\n",
236. setDay.weekSeq, setDay.year % 100, setDay.month,
237. setDay.day, before2.day, before1.day, birthday.weekSeq, birthday.month, birthday.day);
238. }
239. if ( setDay.weekDay == 6 )
240. {
241. printf ("[%02d] %02d.%02d.%02d%10d\n[%02d]%10d %02d.%02d\*\n",
242. setDay.weekSeq, setDay.year % 100, setDay.month,
243. setDay.day, before2.day, birthday.weekSeq, before1.day, birthday.month, birthday.day);
244. }
245. if ( setDay.weekDay == 7 )
246. {
247. printf ("[%02d] %02d.%02d.%02d\n[%02d]%10d%10d %02d.%02d\*\n",
248. setDay.weekSeq, setDay.year % 100, setDay.month,
249. setDay.weekSeq + 1, setDay.day, before2.day, before1.day, birthday.month, birthday.day);
250. }
251. }