

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 | 高梦超 | 学号 | U202413704 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2025.1 | 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

# 实验一

## 2.1实验任务

不同年份秋季校历首周对比

要求改造和升级getMonth getDay getDaySeqOfWeek setYearArray函数

## 2.2实验步骤

关键代码：int getDay( int year, int daySeqOfYear )

{

extern int Month\_LEAP\_YEAR[12];

extern int Month\_NORMAL\_YEAR[12];

int i = 0 ;

int ret = daySeqOfYear ;

if( isLeapYear( year ) == 1 )

{

while( ret > Month\_LEAP\_YEAR[ i ] )

{

ret -= Month\_LEAP\_YEAR[ i ] ;

i ++ ;

}

}

else

{

while( ret > Month\_NORMAL\_YEAR[ i ] )

{

ret -= Month\_NORMAL\_YEAR[ i ] ;

i ++ ;

}

}

return ret ;

}

int getMonth( int year, int dayseq )

{

extern int Month\_LEAP\_YEAR[12];

extern int Month\_NORMAL\_YEAR[12];

int temp = 1 ;

if( isLeapYear( year ) == 1 )

{

while( dayseq > Month\_LEAP\_YEAR[ temp - 1 ] )

{

dayseq -= Month\_LEAP\_YEAR[ temp - 1 ] ;

temp ++ ;

}

}

else

{

while( dayseq > Month\_NORMAL\_YEAR[ temp - 1 ] )

{

dayseq -= Month\_NORMAL\_YEAR[ temp - 1 ] ;

temp ++ ;

}

}

return temp ;

}

int getDaySeqOfWeek( int year, int daySeqOfYear )

{

daySeqOfYear += getDaySeqOnJan1( year ) - 1 ;

daySeqOfYear = daySeqOfYear % 7 ;

return daySeqOfYear ;

}

void setYearArray( int Years[], int yearnum, int year )

{

int num1 ;

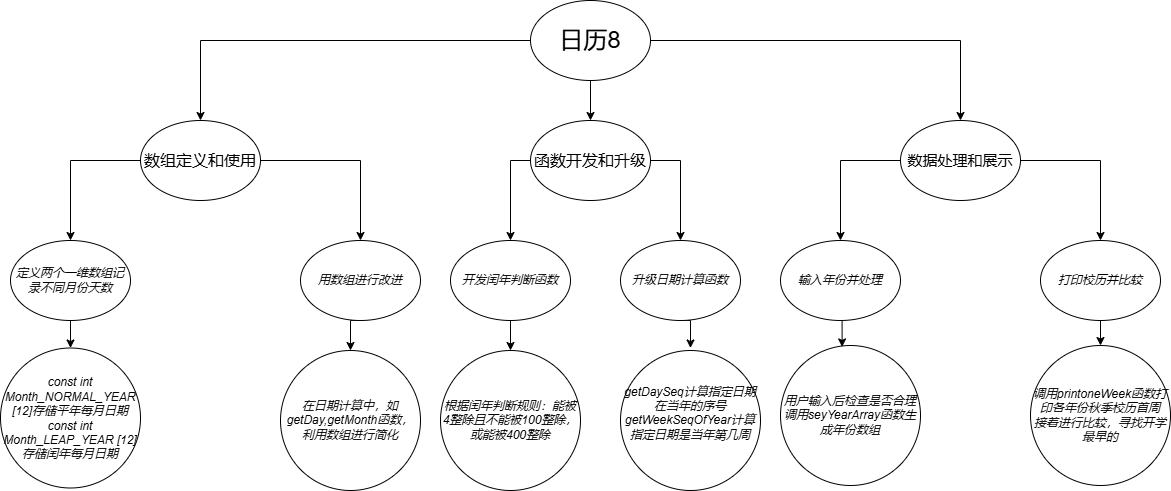
for( num1 = 0; num1 < yearnum ; num1 ++ , year ++)

{

Years[ num1 ] = year;

}

}



## 2.3代码测试

对改进升级后的代码进行测试，检查是否能完成原有功能

## 屏幕截图 2025-01-19 1641252.4实验结论

代码改造成功，能实现原来功能

## 2.5实验总结

定义和初始化数组时，要注意下标是否越界，否则会影响后续程序崩溃，如setYearArray函数

对于逻辑判断要注意，否则可能影响后续的运行，如闰年的判断

# 实验二

## 3.1实验任务

用三维数组实现寻找某人生日的功能

要求改造main函数

## 3.2实验步骤

关键代码：int i,j;

for (i = 0; i < YEAR\_NUM; i ++)

{

int found = 0;

printf("\n%sBirthday in Year %d\n", "", Years[i]);

for (j = 0; j < (isLeapYear(Years[i]) ? 366 : 365); j ++)

{

if (Days[i][j][0] == inputMonth && Days[i][j][1] == inputDay)

{

totalNum ++;

found = 1;

printf("#M:%10s%10s%10s%10s%10s%10s%10s\n",

"Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

printoneWeek(Years[i], getWeekSeqOfYear(Years[i], inputMonth, inputDay), inputMonth);

if (Days[i][j][3] == 6 || Days[i][j][3] == 0)

{

weekendNum++;

}

break;

}

}

if (!found)

{

printf(" Not found.\n");

}

}

printf("\nTotal %d birthdays are found, %d of them are in weekends.\n", totalNum, weekendNum);

return 0;

void printOneDay( int year, int daySeqOfYear , int formatType )

{

int day = getDay( year, daySeqOfYear);

int month = getMonth( year, daySeqOfYear);

int week = getDaySeqOfWeek( year, daySeqOfYear);

if(day <= 0)

{

if( week == 6|| week == 0)

{

printf("%4s%02d.%02d!"," ", 12, day + 31);

}

else

{

printf("%5s%02d.%02d"," ", 12, day + 31);

}

}

else if( formatType == 1 )

{

printf("%2s%02d.%02d.%02d"," ", year % 100 , month, day);

}

else if( day == 1)

{

if( month == 1)

{

if( week == 6|| week == 0)

{

printf("%1s%02d.%02d.%02d!"," ", year % 100 , month, day);

}

else

{

printf("%2s%02d.%02d.%02d"," ", year % 100 , month, day);

}

}

else if( week == 6|| week == 0)

{

printf("%4s%02d.%02d!"," ", month, day);

}

else

{

printf("%5s%02d.%02d"," ", month, day);

}

}

else

{

if( week == 6|| week == 0)

{

printf("%9d!", day);

}

else

{

printf("%10d", day);

}

}

}

void printoneWeek(int year, int weekSeqOfYear, int weekSeqShow )

{

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

int sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - getDaySeqOnJan1( year ) + 2 ;

int sEndSeqOfYear = sStartSeqOfYear + 6 ;

int currentyear = 365 + isLeapYear( year ) ;

int daySeqOfYear = sStartSeqOfYear ;

if( weekSeqShow == 1)

{

printOneDay( year, daySeqOfYear , DATE\_INFO\_FULL ) ;

daySeqOfYear ++ ;

}

for( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )

{

if( daySeqOfYear <= currentyear )

{

printOneDay( year, daySeqOfYear , DATE\_INFO\_BRIEF ) ;

}

else

{

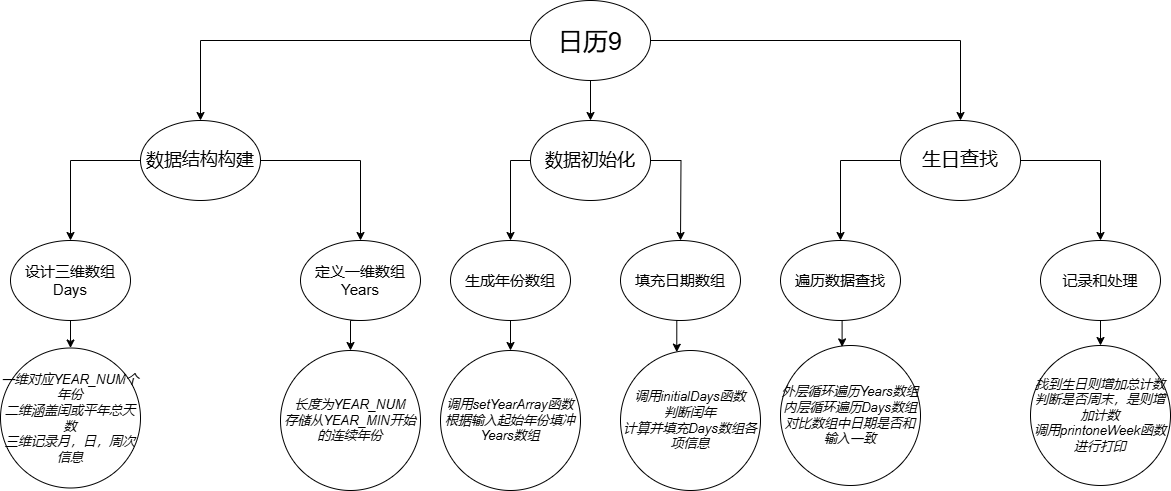
printOneDay( year + 1, daySeqOfYear - currentyear , DATE\_INFO\_BRIEF ) ;

}

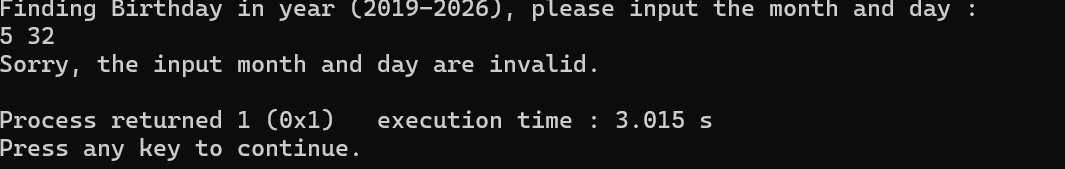
}

printf("\n") ;

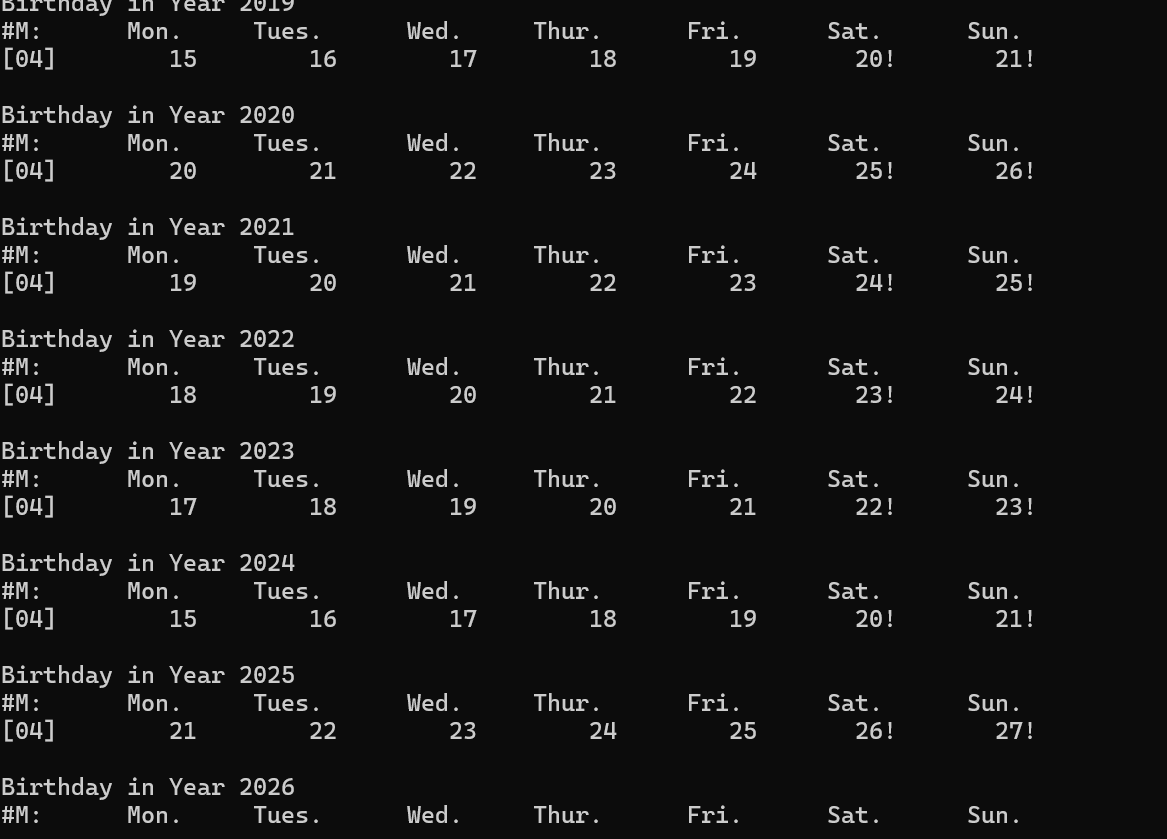
}



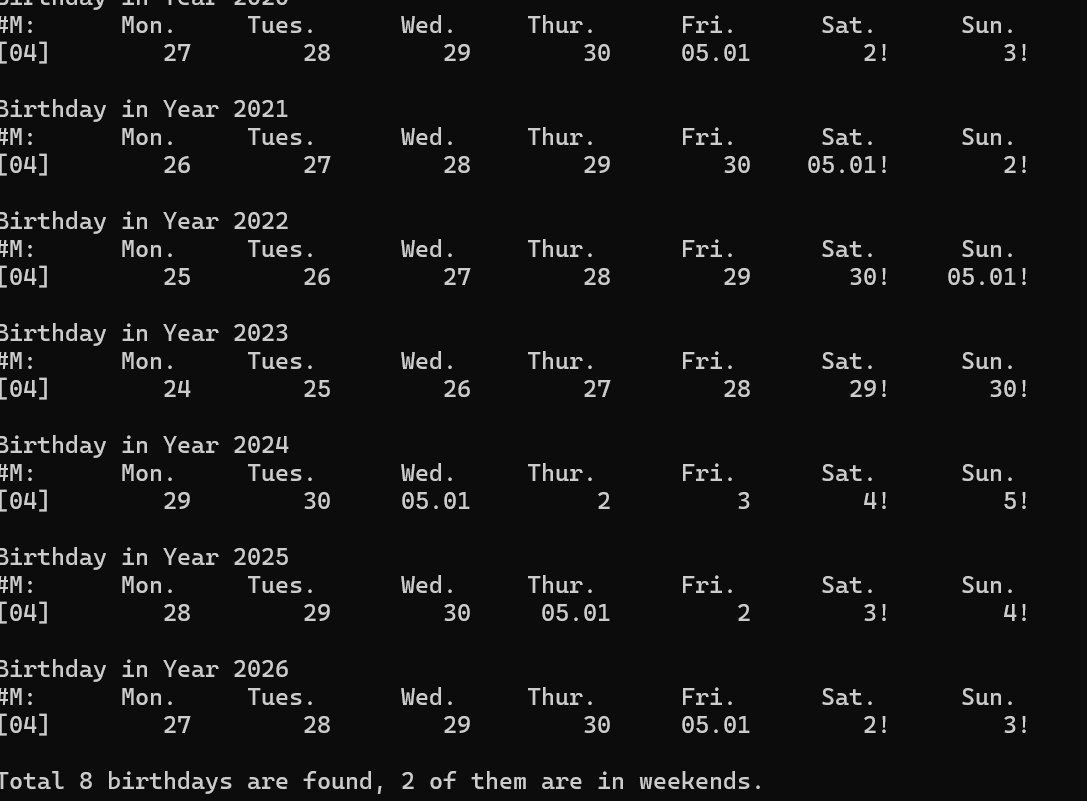
## 2.3代码测试

输入不正确的月或日，看是否会报错

添加打印语句，观看逻辑思维，以及数组中的一些信息是否存在错误



对完整代码进行测试



## 3.4实验结论

代码能够正常运行

## 3.5实验总结

switch语句的使用十分重要，要注意其中的书写规范

代码要有注释，否则合并过程可能会出现看不懂代码的情况（个人经历）

逻辑判断和数组的使用要更加熟悉

# 实验三

## 4.1实验任务

用结构体实现寻找生日

## 4.2实验步骤

关键代码：Day setDay(int year,int month,int day )

{

Day d;

d.year = year;

d.month = month;

d.day = day;

d.daySeq = getDaySeq(year, month, day);//day in a year

d.weekSeq = getWeekSeqOfYear(year, month, day);//week in a year

d.weekDay = getDaySeqOfWeek(year, d.daySeq);//day in a week

return d;

}

Day getDayBefore(Day currentDay , int interval )

{

currentDay.day -= interval;

currentDay.weekDay -= interval;

currentDay.daySeq -= interval;

if(currentDay.weekDay <= 6)

{

currentDay.weekSeq -= (currentDay.weekDay / 7 + 1);

currentDay.weekDay = (currentDay.weekDay % 7 + 7 ) % 7;

}

if(currentDay.day < 1)

{

if(currentDay.month == 1)

{

currentDay.year --;

currentDay.month = 12;

currentDay.daySeq += (isLeapYear(currentDay.year) + 365);

currentDay.day += 31;

currentDay.weekSeq = getWeekSeqOfYear(currentDay.year, currentDay.month, currentDay.day);

}

else

{

currentDay.month --;

if(isLeapYear(currentDay.year) == 0)

{

currentDay.day += monthDayInNormalYear[currentDay.month];

}

else

{

currentDay.day += monthDayInLeapYear[currentDay.month];

}

}

}

return currentDay;

}

Day getDayAfter( Day currentDay, int interval )

{

Day dayAfter;

if(currentDay.daySeq + interval <= isLeapYear(currentDay.year) + 365)

{

dayAfter.year = currentDay.year;

dayAfter.daySeq = currentDay.daySeq + interval;

dayAfter.month = getMonth(dayAfter.year,dayAfter.daySeq);

dayAfter.day = getDay(dayAfter.year,dayAfter.daySeq);

dayAfter.weekSeq = getWeekSeqOfYear(dayAfter.year,dayAfter.month,dayAfter.day);

dayAfter.weekDay = getDaySeqOfWeek(dayAfter.year,dayAfter.daySeq);

}

else

{

dayAfter.year = currentDay.year + 1;

dayAfter.daySeq = currentDay.daySeq + interval - isLeapYear(currentDay.year) - 365;

dayAfter.month = getMonth(dayAfter.year,dayAfter.daySeq);

dayAfter.day = getDay(dayAfter.year,dayAfter.daySeq);

dayAfter.weekSeq = getWeekSeqOfYear(dayAfter.year,dayAfter.month,dayAfter.day);

dayAfter.weekDay = getDaySeqOfWeek(dayAfter.year,dayAfter.daySeq);

}

return dayAfter;

}

int getTwoDaysInterval(Day startDay, Day endDay )

{

if(startDay.year < endDay.year)

{

return -1;

}

else if(startDay.year == endDay.year)

{

if(startDay.daySeq < endDay.daySeq)

{

return -1;

}

else if (startDay.daySeq == endDay.daySeq)

{

return 0;

}

else

{

return 1;

}

}

}

void printDay(Day currentDay, int displayFormat)

{

switch (displayFormat)

{

case 1:

printf("%10s", "");

break;

case 2:

printf("%4d.%02d.%02d", (currentDay.year % 100), currentDay.month, currentDay.day);

break;

case 3:

printf("%10d", currentDay.day);

break;

case 4:

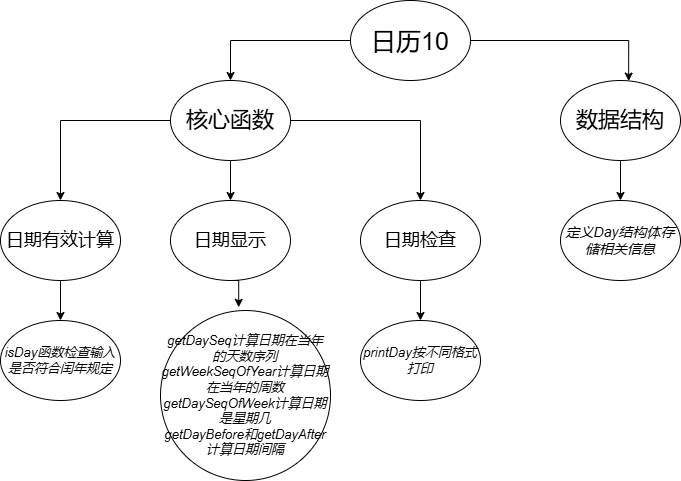
printf(" %02d.%02d\*", currentDay.month, currentDay.day);

break;

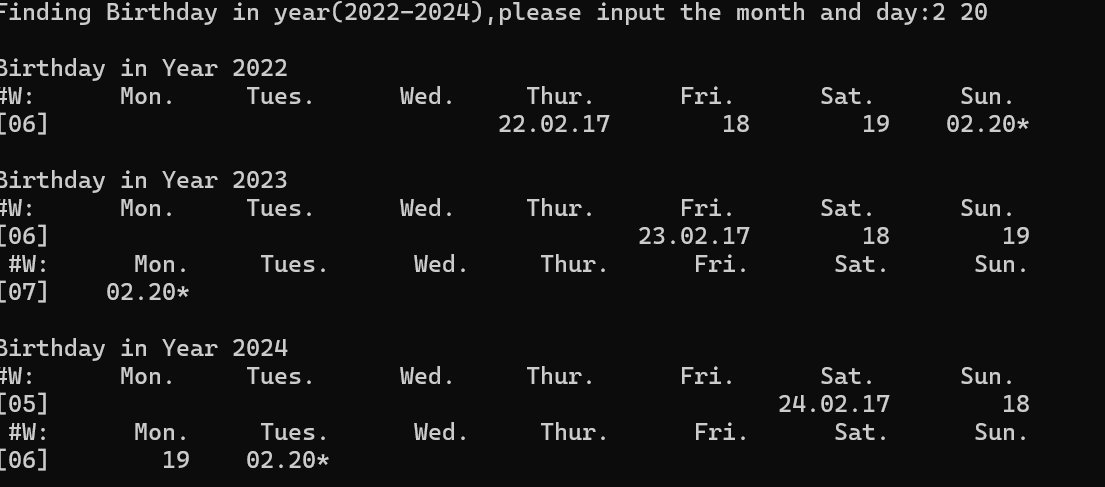
}

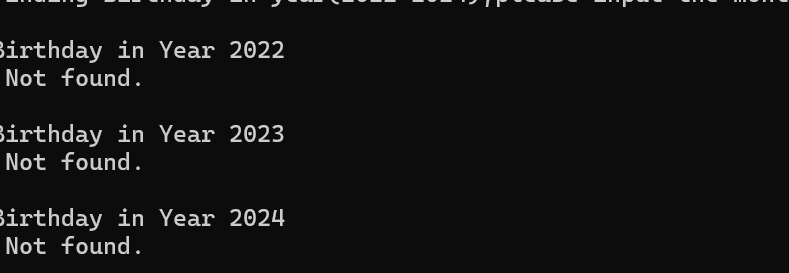
return;

}



## 4.3代码测试





由于是在原基础上通过结构体实现功能，故在此不重复对不分函数进行调试

测试一些特殊情况和正常日期，观察运行结果

## 4.4实验结论

代码能够正常运行

## 4.5实验总结

要熟练掌握结构体的使用

对于一些代码的注释要标注清楚

循环条件要注意

综合使用所学知识

# 课程学习总结

学习代码的是一个比较痛苦的过程，刚刚开始一些简单的逻辑运算和一些语句的使用还相对轻松，随着学习的加深，感受到力不从心，几百行的代码，有时候没思路，上课也听不懂，但是不愿意放弃这门学科，自己努力编代码，课后认真学习，虽然有时候课程作业难度很大，经常卡点完成，或者有时候来不及完成，不过在同学的帮助下，也是一步步的跟上了节奏，虽然不论是机考还是期末，都有一定的难度，恨自己没有全力去学，不过我会在寒假认真学习，补回知识点，总的来说，C语言真是让我又爱又恨！

# 附录

完整实验代码

## 实验一

Main.c

#include <stdio.h>  
#include <stdlib.h>  
#include "date.h"  
#include "funs.h"  
  
**const** **int** Month\_NORMAL\_YEAR [12] =  
{31,28,31,30,31,30,31,31,30,31,30,31};  
  
**const** **int** Month\_LEAP\_YEAR [12] =  
{31,29,31,30,31,30,31,31,30,31,30,31};  
  
**int** **main**()  
{  
    **int** year ,Years[ YEAR\_NUM ];  
  
    **printf**("Spring Calendar Comparison in %d years, please input the first year (%d-%d): \n",  
           YEAR\_NUM, YEAR\_MIN, YEAR\_MAX - YEAR\_NUM ) ;  
  
    **scanf**("%d", &year) ;  
  
    **if**( year < YEAR\_MIN || year + YEAR\_NUM > YEAR\_MAX )  
    {  
        **printf**("\nInvalid input!\n") ;  
        return 0 ;  
    }  
  
    **setYearArray**( Years, YEAR\_NUM, year );  
  
    **int** i;  
    **int** startDays[ YEAR\_NUM ] = { 0 } ;  
  
    **for**( i = 0 ; i< YEAR\_NUM ; i ++ )  
    {  
        **printf**("\n%s%s%d\n"," ","First week in spring calendar of Year ", Years[i]) ;  
        **printf**("#W:%10s%10s%10s%10s%10s%10s%10s\n"  
           ,"Mon.","Tues.","Wend.","Thur.","Fri.","Sat.","Sun.") ;  
        **printoneWeek**( Years[ i ], **getWeekSeqOfYear**( Years[ i ], 9, 4 ), 1) ;  
        startDays[ i ] = **getThisMonday**( Years[ i ], **getDaySeq**( Years[ i ], 9, 4 )) ;  
    }  
  
    **int** min = 0 ;  
  
    **for**( i = 1 ; i < YEAR\_NUM ; i ++)  
    {  
        min = ( startDays[ i ] < startDays[ min ]) ? i : min ;  
    }  
  
    **printf**("\n%s%s%d\n"," " ,"Earliest Spring Semester is in Year " , Years[ min ] );  
  
    return 0 ;  
}

Date.c

#include <stdio.h>  
#include <stdlib.h>  
  
#include "date.h"  
#include "funs.h"  
  
#define MONTH\_NUM 12  
  
**int** **getDay**( **int** year, **int** daySeqOfYear )  
{  
    **extern** **int**  Month\_LEAP\_YEAR[12];  
    **extern** **int**  Month\_NORMAL\_YEAR[12];  
  
    **int** i = 0 ;  
    **int** ret = daySeqOfYear ;  
  
    **if**( **isLeapYear**( year ) == 1 )  
    {  
        **while**( ret > Month\_LEAP\_YEAR[ i ] )  
        {  
            ret -= Month\_LEAP\_YEAR[ i ] ;  
  
            i ++ ;  
        }  
    }  
  
    **else**  
    {  
        **while**( ret > Month\_NORMAL\_YEAR[ i ] )  
        {  
            ret -= Month\_NORMAL\_YEAR[ i ] ;  
  
            i ++ ;  
        }  
    }  
  
    return  ret ;  
}  
  
**int** **getDaySeq**( **int** year, **int** month, **int** day )  
{  
    **switch**( month )  
    {  
        **case** 12:  
            day += 30 ;  
  
        **case** 11:  
            day += 31 ;  
  
        **case** 10:  
            day += 30 ;  
  
        **case** 9:  
            day += 31 ;  
  
        **case** 8:  
            day += 31 ;  
  
        **case** 7:  
            day += 30 ;  
  
        **case** 6:  
            day += 31 ;  
  
        **case** 5:  
            day += 30 ;  
  
        **case** 4:  
            day += 31 ;  
  
        **case** 3:  
            day += 28 + **isLeapYear**( year ) ;  
  
        **case** 2:  
            day += 31 ;  
  
        break ;  
    }  
  
    return day ;  
}  
  
**int** **getDaySeqOfWeek**( **int** year, **int** daySeqOfYear )  
{  
  
    daySeqOfYear += **getDaySeqOnJan1**( year ) - 1 ;  
  
    daySeqOfYear = daySeqOfYear % 7 ;  
  
    return daySeqOfYear ;  
}  
  
**int** **getDaySeqOnJan1**( **int** year )  
{  
    **int** result ;  
  
    result = ( year - 1 +( year - 1) / 4 - ( year - 1)/ 100 +  
             ( year - 1)/ 400) % 7 + 1 ;  
  
    return result ;  
}  
  
**int** **getMonth**( **int** year, **int** dayseq )  
{  
    **extern** **int**  Month\_LEAP\_YEAR[12];  
    **extern** **int**  Month\_NORMAL\_YEAR[12];  
  
    **int** temp = 1 ;  
  
    **if**( **isLeapYear**( year ) == 1 )  
    {  
        **while**( dayseq > Month\_LEAP\_YEAR[ temp - 1 ] )  
        {  
            dayseq -= Month\_LEAP\_YEAR[ temp - 1 ] ;  
  
            temp ++ ;  
        }  
    }  
  
    **else**  
    {  
        **while**( dayseq > Month\_NORMAL\_YEAR[ temp - 1 ] )  
        {  
            dayseq -= Month\_NORMAL\_YEAR[ temp - 1 ] ;  
  
            temp ++ ;  
        }  
    }  
    return temp ;  
}  
  
/\*int getMonthLength( int year , int month )  
{  
    int monthnum = 0 ;  
  
    switch ( month )  
    {  
        case 1:  
        case 3:  
        case 5:  
        case 7:  
        case 8:  
        case 10:  
        case 12:  
  
            monthnum = 31 ;  
            break ;  
  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
  
            monthnum = 30 ;  
            break ;  
  
        case 2:  
  
            monthnum = 28 + isLeapYear( year ) ;  
            break ;  
    }  
  
    return monthnum ;  
}\*/  
  
**int** **getNextMonday**( **int** year, **int** day )  
{  
    **if**( ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 != 1)  
    {  
        **if**(( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 == 0)  
        {  
            day ++ ;  
        }  
  
        **else**  
        {  
            day += (7 -( day + **getDaySeqOnJan1**( year ) - 1 ) % 7) + 1 ;  
        }  
    }  
  
    return day ;  
}  
  
**int** **getThisMonday** ( **int** year, **int** day )  
{  
    **if**( ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 != 1)  
    {  
        **if**(( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 == 0)  
        {  
            day -= 6 ;  
        }  
  
        **else**  
        {  
            day -= ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 - 1 ;  
        }  
    }  
  
    /\* day = getNextMonday ( year, day) - 7 ; \*/  
  
    return day ;  
}  
  
**int** **getThisSunday**( **int** year, **int** day )  
{  
    **if**( ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 != 0)  
    {  
        day += 7 -(day + **getDaySeqOnJan1**( year ) - 1 ) % 7 ;  
    }  
  
    return day ;  
}  
  
**int** **isLeapYear**( **int** year )  
{  
    **if**( ( year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )  
    {  
        return 1 ;  
    }  
  
    return 0 ;  
}  
  
**int** **getWeekSeqOfYear**( **int** year, **int** month, **int** day )  
{  
    **int** week ;  
  
    **int** daySeqOfYear = **getDaySeq**( year, month, day ) ;  
  
    **int** currentyear = 365 + **isLeapYear**( year ) ;  
  
    **if**( daySeqOfYear > currentyear )  
    {  
        **if**( **getDaySeqOfWeek**( year + 1, daySeqOfYear - currentyear) == 0)  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) ) / 7 ;  
        }  
  
        **else**  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) +  
                   ( 7 - **getDaySeqOfWeek**( year + 1, daySeqOfYear - currentyear))) / 7 ;  
        }  
    }  
  
    **else**  
    {  
        **if**( **getDaySeqOfWeek**( year , daySeqOfYear ) == 0)  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) ) / 7 ;  
        }  
  
        **else**  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) +  
                   ( 7 - **getDaySeqOfWeek**( year , daySeqOfYear)))/ 7 ;  
        }  
    }  
  
    return week;  
}  
  
void **setYearArray**( **int** Years[], **int** yearnum, **int** year )  
{  
    **int** num1 ;  
  
    **for**( num1 = 0; num1 < yearnum ; num1 ++ , year ++)  
    {  
        Years[ num1 ] = year;  
    }  
}

Fun.c

#include <stdio.h>  
#include <stdlib.h>  
  
#include "date.h"  
#include "funs.h"  
  
void **printOneDay**( **int** year, **int** daySeqOfYear , **int** formatType )  
{  
    **int** day = **getDay**( year, daySeqOfYear);  
    **int** month = **getMonth**( year, daySeqOfYear);  
    **int** week = **getDaySeqOfWeek**( year, daySeqOfYear);  
  
    **if**( formatType == 1 )  
    {  
        **printf**("%2s%02d.%02d.%02d"," ", year % 100 , month, day);  
    }  
  
    **else** **if**( day == 1)  
    {  
        **if**( month == 1)  
        {  
            **if**( week == 6|| week == 0)  
            {  
                **printf**("%1s%02d.%02d.%02d!"," ", year % 100 , month, day);  
            }  
  
            **else**  
            {  
                **printf**("%2s%02d.%02d.%02d"," ", year % 100 , month, day);  
            }  
        }  
  
        **else** **if**( week == 6|| week == 0)  
        {  
            **printf**("%4s%02d.%02d!"," ", month, day);  
        }  
  
        **else**  
        {  
            **printf**("%5s%02d.%02d"," ", month, day);  
        }  
    }  
  
    **else**  
    {  
        **if**( week == 6|| week == 0)  
        {  
            **printf**("%9d!", day);  
        }  
  
        **else**  
        {  
            **printf**("%10d", day);  
        }  
    }  
}  
  
void **printoneWeek**(**int** year, **int** weekSeqOfYear, **int** weekSeqShow )  
{  
  
    **printf**("[%02d]", weekSeqShow) ;  
  
    **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - **getDaySeqOnJan1**( year ) + 2 ;  
    **int** sEndSeqOfYear = sStartSeqOfYear + 6 ;  
    **int** currentyear = 365 + **isLeapYear**( year ) ;  
    **int** daySeqOfYear = sStartSeqOfYear ;  
  
    **if**( weekSeqShow == 1)  
    {  
        **printOneDay**( year, daySeqOfYear , DATE\_INFO\_FULL ) ;  
  
        daySeqOfYear ++ ;  
    }  
  
  
    **for**( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )  
    {  
        **if**( daySeqOfYear <= currentyear )  
        {  
            **printOneDay**( year, daySeqOfYear , DATE\_INFO\_BRIEF ) ;  
        }  
  
        **else**  
        {  
            **printOneDay**( year + 1, daySeqOfYear - currentyear , DATE\_INFO\_BRIEF ) ;  
        }  
  
    }  
  
    **printf**("\n") ;  
}

Date.h

#ifndef DATE\_H\_INCLUDED  
#define DATE\_H\_INCLUDED  
  
#define YEAR\_MIN 2000  
#define YEAR\_MAX 2030  
#define YEAR\_NUM 5  
#define DATE\_INFO\_BRIEF 0  
#define DATE\_INFO\_FULL 1  
  
**int** **isLeapYear**( **int** year );  
**int** **getDaySeqOnJan1**( **int** year );  
  
/\*int getMonthLength( int year , int month );\*/  
**int** **getDaySeq**( **int** year, **int** month, **int** day );  
**int** **getWeekSeqOfYear**( **int** year, **int** month, **int** day );  
  
**int** **getMonth**( **int** year, **int** daySeqOfYear );  
**int** **getDay**( **int** year, **int** daySeqOfYear );  
**int** **getDaySeqOfWeek**( **int** year,**int** daySeqOfYear );  
  
**int** **getNextMonday**( **int** year, **int** daySeqOfYear );  
**int** **getThisMonday** ( **int** year, **int** day );  
**int** **getThisSunday**( **int** year, **int** daySeqOfYear );  
  
void **setYearArray**( **int** Years[], **int** yearnum, **int** year );  
  
#endif // DATE\_H\_INCLUDED

Fun.h

#ifndef FUNS\_H\_INCLUDED  
#define FUNS\_H\_INCLUDED  
  
void **printOneDay**( **int** year, **int** daySeqOfYear , **int** formatType );  
void **printoneWeek**(**int** year, **int** weekSeqOfYear, **int** weekSeqShow );  
  
#endif // FUNS\_H\_INCLUDED

## 实验二

Main.c

#include <stdio.h>  
#include <stdlib.h>  
#include "date.h"  
#include "funs.h"  
  
**const** **int** Month\_NORMAL\_YEAR [12] =  
{31,28,31,30,31,30,31,31,30,31,30,31};  
  
**const** **int** Month\_LEAP\_YEAR [12] =  
{31,29,31,30,31,30,31,31,30,31,30,31};  
  
  
**int** **main**()  
{  
    //Declare variables  
    **int** inputMonth, inputDay, Years[YEAR\_NUM];  
    //Declare a three-dimension array to record date information.int Days[YEAR\_NUM][366][4]={0]};  
    // D1splay the program information  
    **printf**("Finding Birthday in year (%d-%d), please input the month and day : \n", YEAR\_MIN, YEAR\_MAX );  
  
    **scanf**("%d%d", &inputMonth, &inputDay );  
  
    **if** ( inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > 31)  
    {  
        **printf**("Sorry, the input month and day are invalid.\n");  
        return 1;  
    }  
  
    **setYearArray** ( Years, YEAR\_NUM, YEAR\_MIN );  
  
    **int** Days[YEAR\_NUM][366][4] = {0};  
  
    **initialDays** ( Years, Days, YEAR\_NUM);  
  
    **int** totalNum = 0;  
    **int** weekendNum = 0;  
  
    //Requirement: finish the rest functions  
    // 1.   the print related sentences:  
    //      printf("\n%s%sNd\n","","Birthday in Year ", year );  
    //      printf("#M:%10s%10s%10s%10s%10s%105%10s\n",  
    //      "Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.");  
    //      printf(" Not found.\n");  
    // 2.   the print related function calling:// printOneWeek (year, weekSeq, month);/  
    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
    //<< Your source code starts here >>//  
    //  
    **int** i,j;  
    **for** (i = 0; i < YEAR\_NUM; i ++)  
    {  
        **int** found = 0;  
        **printf**("\n%sBirthday in Year %d\n", "", Years[i]);  
  
        **for** (j = 0; j < (**isLeapYear**(Years[i]) ? 366 : 365); j ++)  
        {  
            **if** (Days[i][j][0] == inputMonth && Days[i][j][1] == inputDay)  
            {  
                totalNum ++;  
                found = 1;  
                **printf**("#M:%10s%10s%10s%10s%10s%10s%10s\n",  
                       "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");  
  
                **printoneWeek**(Years[i], **getWeekSeqOfYear**(Years[i], inputMonth, inputDay), inputMonth);  
  
                **if** (Days[i][j][3] == 6 || Days[i][j][3] == 0)  
                {  
                    weekendNum++;  
                }  
                break;  
            }  
        }  
  
        **if** (!found)  
        {  
            **printf**(" Not found.\n");  
        }  
    }  
    **printf**("\nTotal %d birthdays are found, %d of them are in weekends.\n", totalNum, weekendNum);  
    return 0;  
}  
  
  
  
  
void **initialDays**(**int** Years[], **int** Days[][366][4], **int** yearNum )  
{  
    **int** year, month, day, yearLength, weekSeq, seqOfWeek;  
    **int** i, j;  
  
    **for** (i = 0; i < YEAR\_NUM; i ++)  
    {  
        year = Years[i];  
        yearLength = **isLeapYear**( year ) ? 366 : 365;  
        **for**(j = 0; j < yearLength; j ++)  
        {  
        Days[i][j][0] = **getMonth**( year, j + 1);  
        Days[i][j][1] = **getDay**( year, j + 1 );  
        Days[i][j][2] = **getWeekSeqOfYear**( year, Days[i][j][0], Days[i][j][1] );  
        Days[i][j][3] = **getDaySeqOfWeek**( year, j + 1 );  
        }  
    }  
}

Date.c

#include <stdio.h>  
#include <stdlib.h>  
  
#include "date.h"  
#include "funs.h"  
  
#define MONTH\_NUM 12  
  
**int** **getDay**( **int** year, **int** daySeqOfYear )  
{  
    **extern** **int**  Month\_LEAP\_YEAR[12];  
    **extern** **int**  Month\_NORMAL\_YEAR[12];  
  
    **int** i = 0 ;  
    **int** ret = daySeqOfYear ;  
  
    **if**( **isLeapYear**( year ) == 1 )  
    {  
        **while**( ret > Month\_LEAP\_YEAR[ i ] )  
        {  
            ret -= Month\_LEAP\_YEAR[ i ] ;  
  
            i ++ ;  
        }  
    }  
  
    **else**  
    {  
        **while**( ret > Month\_NORMAL\_YEAR[ i ] )  
        {  
            ret -= Month\_NORMAL\_YEAR[ i ] ;  
  
            i ++ ;  
        }  
    }  
  
    return  ret ;  
}  
  
**int** **getDaySeq**( **int** year, **int** month, **int** day )  
{  
    **switch**( month )  
    {  
        **case** 12:  
            day += 30 ;  
  
        **case** 11:  
            day += 31 ;  
  
        **case** 10:  
            day += 30 ;  
  
        **case** 9:  
            day += 31 ;  
  
        **case** 8:  
            day += 31 ;  
  
        **case** 7:  
            day += 30 ;  
  
        **case** 6:  
            day += 31 ;  
  
        **case** 5:  
            day += 30 ;  
  
        **case** 4:  
            day += 31 ;  
  
        **case** 3:  
            day += 28 + **isLeapYear**( year ) ;  
  
        **case** 2:  
            day += 31 ;  
  
        break ;  
    }  
  
    return day ;  
}  
  
**int** **getDaySeqOfWeek**( **int** year, **int** daySeqOfYear )  
{  
  
    daySeqOfYear += **getDaySeqOnJan1**( year ) - 1 ;  
  
    daySeqOfYear = daySeqOfYear % 7 ;  
  
    return daySeqOfYear ;  
}  
  
**int** **getDaySeqOnJan1**( **int** year )  
{  
    **int** result ;  
  
    result = ( year - 1 +( year - 1) / 4 - ( year - 1)/ 100 +( year - 1)/ 400) % 7 + 1 ;  
  
    return result ;  
}  
  
**int** **getMonth**( **int** year, **int** dayseq )  
{  
    **extern** **int**  Month\_LEAP\_YEAR[12];  
    **extern** **int**  Month\_NORMAL\_YEAR[12];  
  
    **int** temp = 1 ;  
  
    **if**( **isLeapYear**( year ) == 1 )  
    {  
        **while**( dayseq > Month\_LEAP\_YEAR[ temp - 1 ] )  
        {  
            dayseq -= Month\_LEAP\_YEAR[ temp - 1 ] ;  
  
            temp ++ ;  
        }  
    }  
  
    **else**  
    {  
        **while**( dayseq > Month\_NORMAL\_YEAR[ temp - 1 ] )  
        {  
            dayseq -= Month\_NORMAL\_YEAR[ temp - 1 ] ;  
  
            temp ++ ;  
        }  
    }  
    return temp ;  
}  
  
/\*int getMonthLength( int year , int month )  
{  
    int monthnum = 0 ;  
  
    switch ( month )  
    {  
        case 1:  
        case 3:  
        case 5:  
        case 7:  
        case 8:  
        case 10:  
        case 12:  
  
            monthnum = 31 ;  
            break ;  
  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
  
            monthnum = 30 ;  
            break ;  
  
        case 2:  
  
            monthnum = 28 + isLeapYear( year ) ;  
            break ;  
    }  
  
    return monthnum ;  
}\*/  
  
**int** **getNextMonday**( **int** year, **int** day )  
{  
    **if**( ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 != 1)  
    {  
        **if**(( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 == 0)  
        {  
            day ++ ;  
        }  
  
        **else**  
        {  
            day += (7 -( day + **getDaySeqOnJan1**( year ) - 1 ) % 7) + 1 ;  
        }  
    }  
  
    return day ;  
}  
  
**int** **getThisMonday** ( **int** year, **int** day )  
{  
    **if**( ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 != 1)  
    {  
        **if**(( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 == 0)  
        {  
            day -= 6 ;  
        }  
  
        **else**  
        {  
            day -= ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 - 1 ;  
        }  
    }  
  
    /\* day = getNextMonday ( year, day) - 7 ; \*/  
  
    return day ;  
}  
  
**int** **getThisSunday**( **int** year, **int** day )  
{  
    **if**( ( day + **getDaySeqOnJan1**( year ) - 1 ) % 7 != 0)  
    {  
        day += 7 -(day + **getDaySeqOnJan1**( year ) - 1 ) % 7 ;  
    }  
  
    return day ;  
}  
  
**int** **isLeapYear**( **int** year )  
{  
    **if**( ( year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )  
    {  
        return 1 ;  
    }  
  
    return 0 ;  
}  
  
**int** **getWeekSeqOfYear**( **int** year, **int** month, **int** day )  
{  
    **int** week ;  
  
    **int** daySeqOfYear = **getDaySeq**( year, month, day ) ;  
  
    **int** currentyear = 365 + **isLeapYear**( year ) ;  
  
    **if**( daySeqOfYear > currentyear )  
    {  
        **if**( **getDaySeqOfWeek**( year + 1, daySeqOfYear - currentyear) == 0)  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) ) / 7 ;  
        }  
  
        **else**  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) +  
                   ( 7 - **getDaySeqOfWeek**( year + 1, daySeqOfYear - currentyear))) / 7 ;  
        }  
    }  
  
    **else**  
    {  
        **if**( **getDaySeqOfWeek**( year , daySeqOfYear ) == 0)  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) ) / 7 ;  
        }  
  
        **else**  
        {  
            week = ( daySeqOfYear + **getDaySeqOnJan1**( year ) +  
                   ( 7 - **getDaySeqOfWeek**( year , daySeqOfYear)))/ 7 ;  
        }  
    }  
  
    return week;  
}  
  
void **setYearArray**( **int** Years[], **int** yearnum, **int** year )  
{  
    **int** num1 ;  
  
    **for**( num1 = 0; num1 < yearnum ; num1 ++ , year ++)  
    {  
        Years[ num1 ] = year;  
    }  
}

Fun.c

#include <stdio.h>  
#include <stdlib.h>  
  
#include "date.h"  
#include "funs.h"  
  
void **printOneDay**( **int** year, **int** daySeqOfYear , **int** formatType )  
{  
    **int** day = **getDay**( year, daySeqOfYear);  
    **int** month = **getMonth**( year, daySeqOfYear);  
    **int** week = **getDaySeqOfWeek**( year, daySeqOfYear);  
  
    **if**(day <= 0)  
    {  
        **if**( week == 6|| week == 0)  
        {  
            **printf**("%4s%02d.%02d!"," ", 12, day + 31);  
        }  
  
        **else**  
        {  
            **printf**("%5s%02d.%02d"," ", 12, day + 31);  
        }  
    }  
    **else** **if**( formatType == 1 )  
    {  
        **printf**("%2s%02d.%02d.%02d"," ", year % 100 , month, day);  
    }  
    **else** **if**( day == 1)  
    {  
        **if**( month == 1)  
        {  
            **if**( week == 6|| week == 0)  
            {  
                **printf**("%1s%02d.%02d.%02d!"," ", year % 100 , month, day);  
            }  
  
            **else**  
            {  
                **printf**("%2s%02d.%02d.%02d"," ", year % 100 , month, day);  
            }  
        }  
  
        **else** **if**( week == 6|| week == 0)  
        {  
            **printf**("%4s%02d.%02d!"," ", month, day);  
        }  
  
        **else**  
        {  
            **printf**("%5s%02d.%02d"," ", month, day);  
        }  
    }  
  
    **else**  
    {  
        **if**( week == 6|| week == 0)  
        {  
            **printf**("%9d!", day);  
        }  
  
        **else**  
        {  
            **printf**("%10d", day);  
        }  
    }  
}  
  
void **printoneWeek**(**int** year, **int** weekSeqOfYear, **int** weekSeqShow )  
{  
  
  
    **printf**("[%02d]", weekSeqShow) ;  
  
    **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - **getDaySeqOnJan1**( year ) + 2 ;  
    **int** sEndSeqOfYear = sStartSeqOfYear + 6 ;  
    **int** currentyear = 365 + **isLeapYear**( year ) ;  
    **int** daySeqOfYear = sStartSeqOfYear ;  
  
    **if**( weekSeqShow == 1)  
    {  
        **printOneDay**( year, daySeqOfYear , DATE\_INFO\_FULL ) ;  
  
        daySeqOfYear ++ ;  
    }  
  
  
    **for**( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )  
    {  
        **if**( daySeqOfYear <= currentyear )  
        {  
            **printOneDay**( year, daySeqOfYear , DATE\_INFO\_BRIEF ) ;  
        }  
  
        **else**  
        {  
            **printOneDay**( year + 1, daySeqOfYear - currentyear , DATE\_INFO\_BRIEF ) ;  
        }  
  
    }  
  
    **printf**("\n") ;  
}

Date.h

#ifndef DATE\_H\_INCLUDED  
#define DATE\_H\_INCLUDED  
  
#define YEAR\_MIN 2019  
#define YEAR\_MAX 2026  
#define YEAR\_NUM 8  
#define DATE\_INFO\_BRIEF 0  
#define DATE\_INFO\_FULL 1  
  
**int** **isLeapYear**( **int** year );  
**int** **getDaySeqOnJan1**( **int** year );  
  
/\*int getMonthLength( int year , int month );\*/  
**int** **getDaySeq**( **int** year, **int** month, **int** day );  
**int** **getWeekSeqOfYear**( **int** year, **int** month, **int** day );  
  
**int** **getMonth**( **int** year, **int** daySeqOfYear );  
**int** **getDay**( **int** year, **int** daySeqOfYear );  
**int** **getDaySeqOfWeek**( **int** year,**int** daySeqOfYear );  
  
**int** **getNextMonday**( **int** year, **int** daySeqOfYear );  
**int** **getThisMonday** ( **int** year, **int** day );  
**int** **getThisSunday**( **int** year, **int** daySeqOfYear );  
  
void **setYearArray**( **int** Years[], **int** yearnum, **int** year );  
  
#endif // DATE\_H\_INCLUDED

Fun.h

#ifndef FUNS\_H\_INCLUDED  
#define FUNS\_H\_INCLUDED  
  
void **printOneDay**( **int** year, **int** daySeqOfYear , **int** formatType );  
void **printoneWeek**(**int** year, **int** weekSeqOfYear, **int** weekSeqShow );  
  
#endif // FUNS\_H\_INCLUDED

## 实验三

Main.c

#include <stdio.h>

#include <stdlib.h>

#define YEAR\_NUM 3

#define DATE\_NOSHOW 1

#define DATE\_INFO\_FULL 2

#define DATE\_INFO\_BRIEF 3

#define DATE\_STAR 4

const int monthDayInNormalYear[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

const int monthDayInLeapYear[13] = {0,31,29,31,30,31,30,31,31,30,31,30,31};

const int Years[YEAR\_NUM] = {2022,2023,2024};

const int printDayRange = 3;

typedef struct

{

int year;

int daySeq;

int month;

int day;

int weekSeq;

int weekDay;

}Day;

//Functions for years

int isLeapYear(int year);

int getDaySeqOnJan1(int year);

//Functions for information of dates

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

int getWeekSeqOfYear(int year, int month, int day);

int getMonth(int year, int daySeqOfYear);

int getDay(int year,int daySeqOfYear);

int getDaySeqOfWeek(int year, int daySeqOfYear);

//Functions in main

int isDay(int year, int month, int day );

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

Day getDayBefore(Day currentDay , int interval );

Day getDayAfter( Day currentDay, int interval );

int getTwoDaysInterval(Day startDay, Day endDay );

//Function As a printer

void printDay(Day currentDay, int displayFormat);

int main()

{

int inputMonth;

int inputDay;

int displayFormat;

Day birthDay;

Day prepareDay;

Day printStartDay;

Day currentDay;

printf("Finding Birthday in year(2022-2024),please input the month and day:");

scanf("%d%d", &inputMonth, &inputDay);

int i;

for ( i = 0 ;i < YEAR\_NUM ; i ++ )

{

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

if( isDay(Years[i], inputMonth, inputDay) != 1)

{

printf(" Not found.\n");

continue;

}

birthDay = setDay (Years[i], inputMonth, inputDay );

prepareDay = getDayBefore( birthDay, printDayRange );

printStartDay = getDayBefore( prepareDay, prepareDay.weekDay );

printf("#W:%10s%10s%10s%10s%10s%10s%10s\n[%02d]",

"Mon.", "Tues.","Wed.","Thur.","Fri.","Sat.","Sun.",printStartDay.weekSeq);

for (currentDay = printStartDay; getTwoDaysInterval(currentDay,birthDay) <= 0 ; currentDay = getDayAfter( currentDay , 1 ))

{

if ( getTwoDaysInterval(currentDay, prepareDay ) < 0)

{

displayFormat = DATE\_NOSHOW;

}

else if (getTwoDaysInterval(currentDay, prepareDay ) == 0)

{

displayFormat = DATE\_INFO\_FULL;

}

else if ( getTwoDaysInterval(currentDay, birthDay ) < 0 )

{

displayFormat = DATE\_INFO\_BRIEF;

}

else if ( getTwoDaysInterval(currentDay, birthDay ) == 0)

{

displayFormat = DATE\_STAR;

}

printDay(currentDay, displayFormat);

if(currentDay.weekDay == 6 && displayFormat != DATE\_STAR)

{

printf("\n #W:%10s%10s%10s%10s%10s%10s%10s\n[%02d]",

"Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.",printStartDay.weekSeq + 1);

}

}

printf("\n");

}

return 0;

}

int isLeapYear(int year)

{

if((year % 4 == 0 && year % 100 != 0 ) || year % 400 == 0)

{

return 1;

}

else

{

return 0;

}

}

int getDaySeqOnJan1(int year)

{

int day = 1;

while (year > 2001)

{

day += (isLeapYear(year - 1) + 365);

year --;

}

int SeqOnJan1 = (day - 1) % 7 - 1;

return SeqOnJan1;

}

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

{

int daySeq = day;

int i;

for( i = 1 ; i < month ; i ++)

{

if(isLeapYear(year) == 1)

{

daySeq += monthDayInLeapYear[i];

}

else

{

daySeq += monthDayInNormalYear[i];

}

}

return daySeq;

}

int getWeekSeqOfYear(int year, int month, int day)

{

return (getDaySeq( year , month , day ) + getDaySeqOnJan1(year) - 2) / 7 + 1;

}

int getMonth(int year, int daySeqOfYear)

{

int month\_num;

if(isLeapYear(year) == 1)

{

for(month\_num = 1 ; daySeqOfYear > monthDayInLeapYear[month\_num] ; month\_num ++ )

{

daySeqOfYear -= monthDayInLeapYear[month\_num];

}

}

else if(isLeapYear(year) == 0)

{

for(month\_num = 1 ; daySeqOfYear > monthDayInNormalYear[month\_num] ; month\_num ++ )

{

daySeqOfYear -= monthDayInNormalYear[month\_num];

}

}

return month\_num;

}

int getDay(int year,int daySeqOfYear)

{

int month\_num;

if(isLeapYear(year) == 1)

{

for(month\_num = 1 ; daySeqOfYear > monthDayInLeapYear[month\_num] ; month\_num ++ )

{

daySeqOfYear -= monthDayInLeapYear[month\_num];

}

}

else if(isLeapYear(year) == 0)

{

for(month\_num = 1 ; daySeqOfYear > monthDayInNormalYear[month\_num] ; month\_num ++ )

{

daySeqOfYear -= monthDayInNormalYear[month\_num];

}

}

return daySeqOfYear;

}

int getDaySeqOfWeek(int year, int daySeqOfYear)

{

int Seq;

Seq = (getDaySeqOnJan1(year) + daySeqOfYear) % 7;

return Seq;

}

int isDay(int year, int month, int day )

{

if((month < 0 || month > 12) || day < 0)

{

return 0;

}

else if(isLeapYear(year) == 1 && day > monthDayInLeapYear[month])

{

return 0;

}

else if(isLeapYear(year) == 0 && day > monthDayInNormalYear[month])

{

return 0;

}

else

{

return 1;

}

}

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

{

Day d;

d.year = year;

d.month = month;

d.day = day;

d.daySeq = getDaySeq(year, month, day);//day in a year

d.weekSeq = getWeekSeqOfYear(year, month, day);//week in a year

d.weekDay = getDaySeqOfWeek(year, d.daySeq);//day in a week

return d;

}

Day getDayBefore(Day currentDay , int interval )

{

currentDay.day -= interval;

currentDay.weekDay -= interval;

currentDay.daySeq -= interval;

if(currentDay.weekDay <= 6)

{

currentDay.weekSeq -= (currentDay.weekDay / 7 + 1);

currentDay.weekDay = (currentDay.weekDay % 7 + 7 ) % 7;

}

if(currentDay.day < 1)

{

if(currentDay.month == 1)

{

currentDay.year --;

currentDay.month = 12;

currentDay.daySeq += (isLeapYear(currentDay.year) + 365);

currentDay.day += 31;

currentDay.weekSeq = getWeekSeqOfYear(currentDay.year, currentDay.month, currentDay.day);

}

else

{

currentDay.month --;

if(isLeapYear(currentDay.year) == 0)

{

currentDay.day += monthDayInNormalYear[currentDay.month];

}

else

{

currentDay.day += monthDayInLeapYear[currentDay.month];

}

}

}

return currentDay;

}

Day getDayAfter( Day currentDay, int interval )

{

Day dayAfter;

if(currentDay.daySeq + interval <= isLeapYear(currentDay.year) + 365)

{

dayAfter.year = currentDay.year;

dayAfter.daySeq = currentDay.daySeq + interval;

dayAfter.month = getMonth(dayAfter.year,dayAfter.daySeq);

dayAfter.day = getDay(dayAfter.year,dayAfter.daySeq);

dayAfter.weekSeq = getWeekSeqOfYear(dayAfter.year,dayAfter.month,dayAfter.day);

dayAfter.weekDay = getDaySeqOfWeek(dayAfter.year,dayAfter.daySeq);

}

else

{

dayAfter.year = currentDay.year + 1;

dayAfter.daySeq = currentDay.daySeq + interval - isLeapYear(currentDay.year) - 365;

dayAfter.month = getMonth(dayAfter.year,dayAfter.daySeq);

dayAfter.day = getDay(dayAfter.year,dayAfter.daySeq);

dayAfter.weekSeq = getWeekSeqOfYear(dayAfter.year,dayAfter.month,dayAfter.day);

dayAfter.weekDay = getDaySeqOfWeek(dayAfter.year,dayAfter.daySeq);

}

return dayAfter;

}

int getTwoDaysInterval(Day startDay, Day endDay )

{

if(startDay.year < endDay.year)

{

return -1;

}

else if(startDay.year == endDay.year)

{

if(startDay.daySeq < endDay.daySeq)

{

return -1;

}

else if (startDay.daySeq == endDay.daySeq)

{

return 0;

}

else

{

return 1;

}

}

}

void printDay(Day currentDay, int displayFormat)

{

switch (displayFormat)

{

case 1:

printf("%10s", "");

break;

case 2:

printf("%4d.%02d.%02d", (currentDay.year % 100), currentDay.month, currentDay.day);

break;

case 3:

printf("%10d", currentDay.day);

break;

case 4:

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

break;

}

return;

}