

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 | 袁浩然 | 学号 | U202413677 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2024.12 | 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（日历系列、大数计算系列，选择其中之一）。

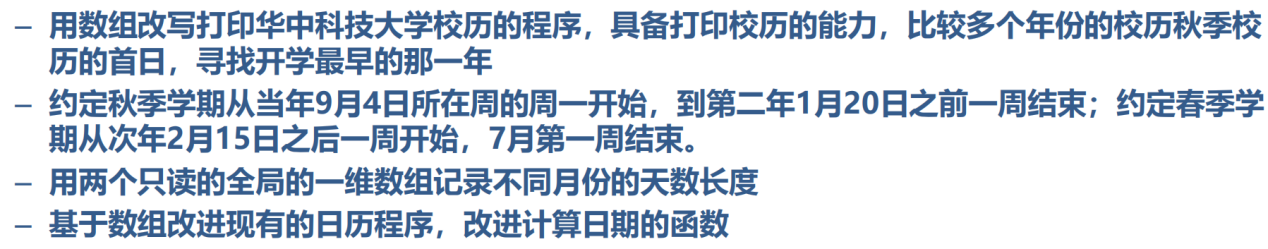
# 实验环境

操作系统：Windows 10

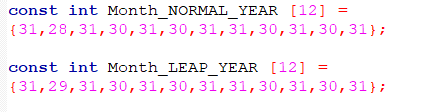
编程工具：CodeBlocks 16.01

# 实验一

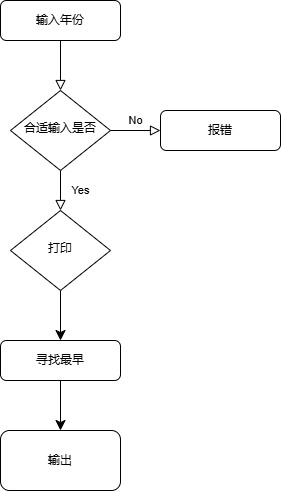
## 实验任务



## 实验步骤



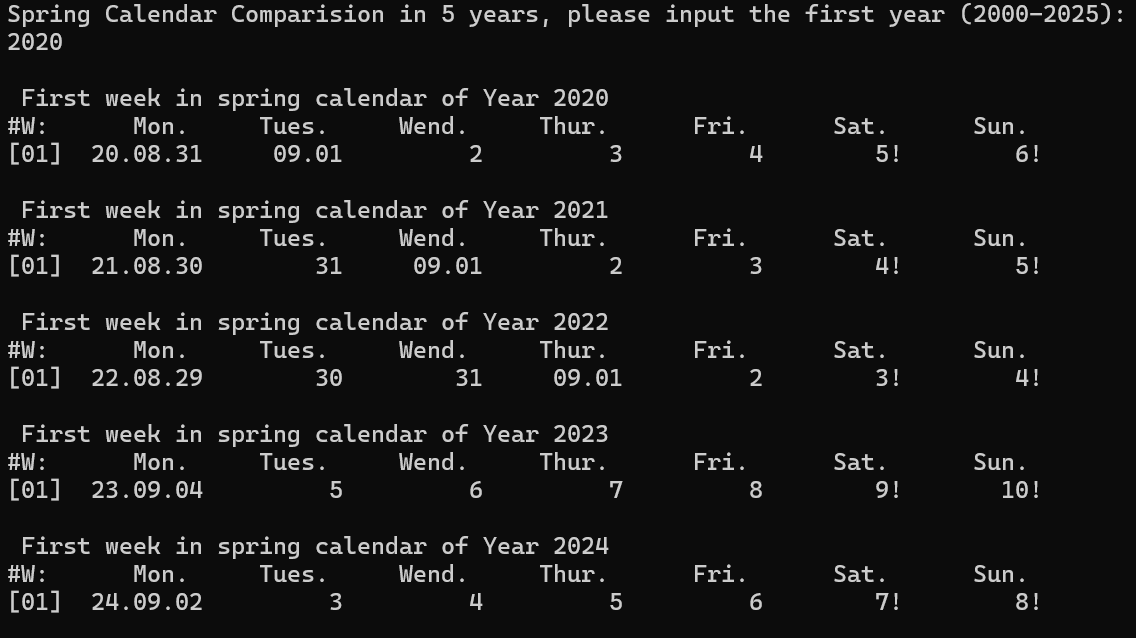
在程序开头设置两个只读数组



## 代码测试

### 测试点 xxx的测试结果

输入年份2020



## 实验结论

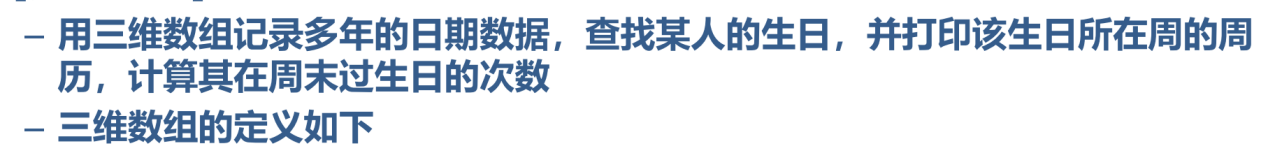
代码达到功能目标

## 实验总结

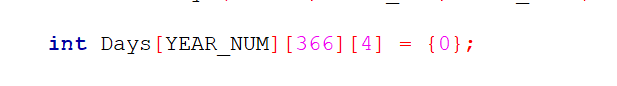
实验主要是换月需要处理一下，然后调用以前的函数还算比较顺利。

# 实验二

4.1实验任务

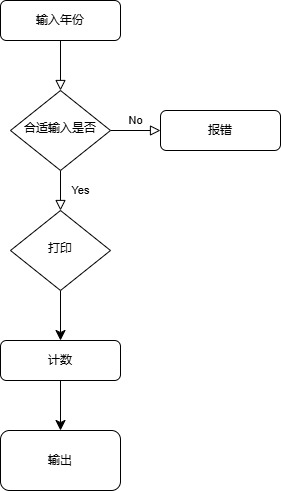


## 4.2实验步骤



先定义一个多维数组

然后按以下步骤



#### 4.3 代码测试

输入特殊值 2 29测试



#### 4.4实验结论

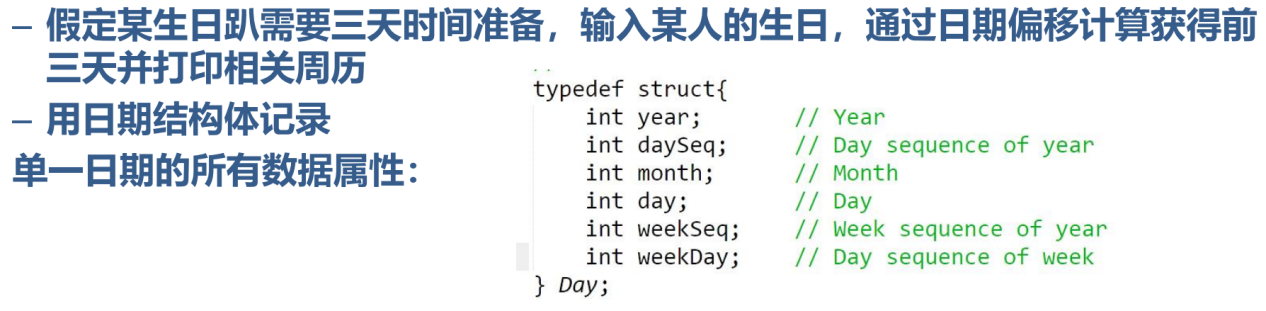
测试通过，达到目标

#### 4.5实验总结

代码的关键在于处理年月的问题，学会在函数中增加判断就会减少很多重复部分。

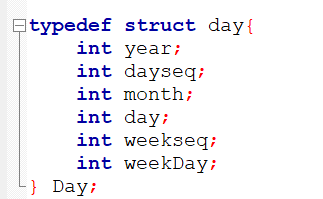
# 实验三

#### 5.1实验任务

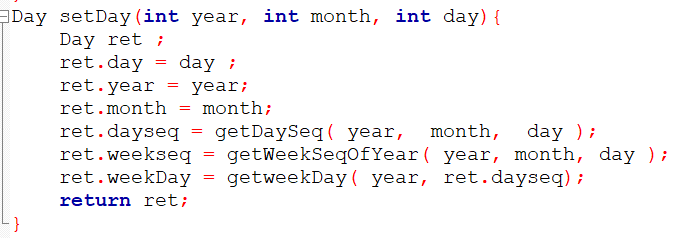


#### 5.2实验步骤

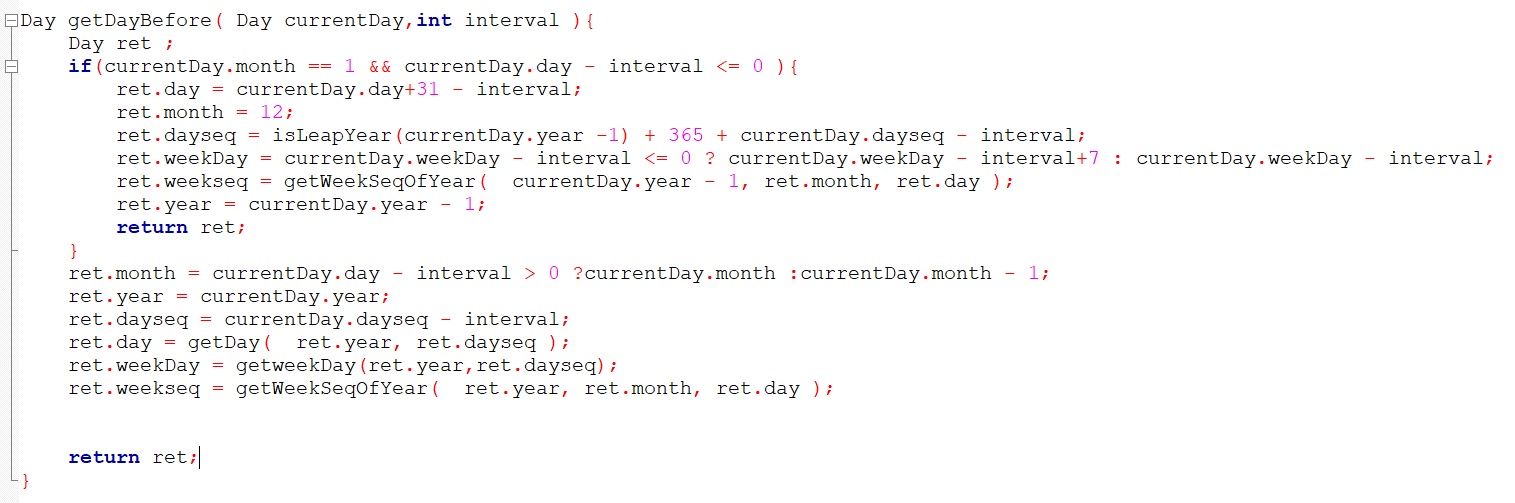
先定义一个结构体

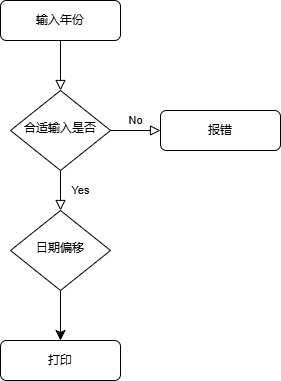


设置新的结构体函数



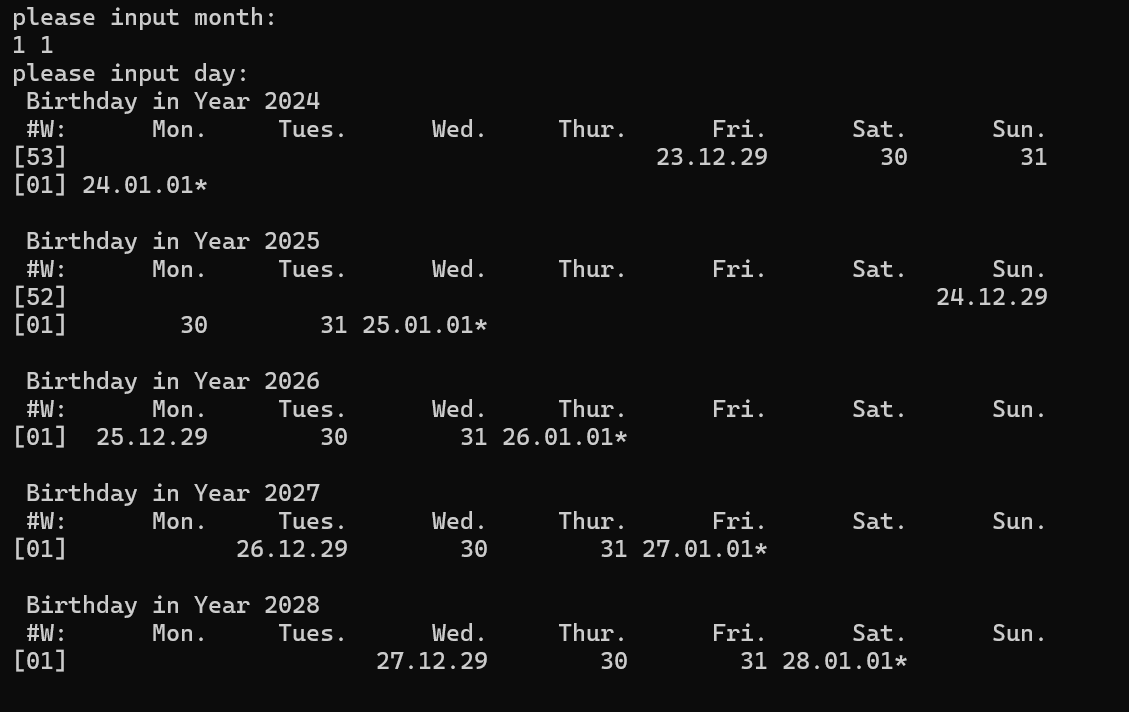
再设置函数寻找之前的日期



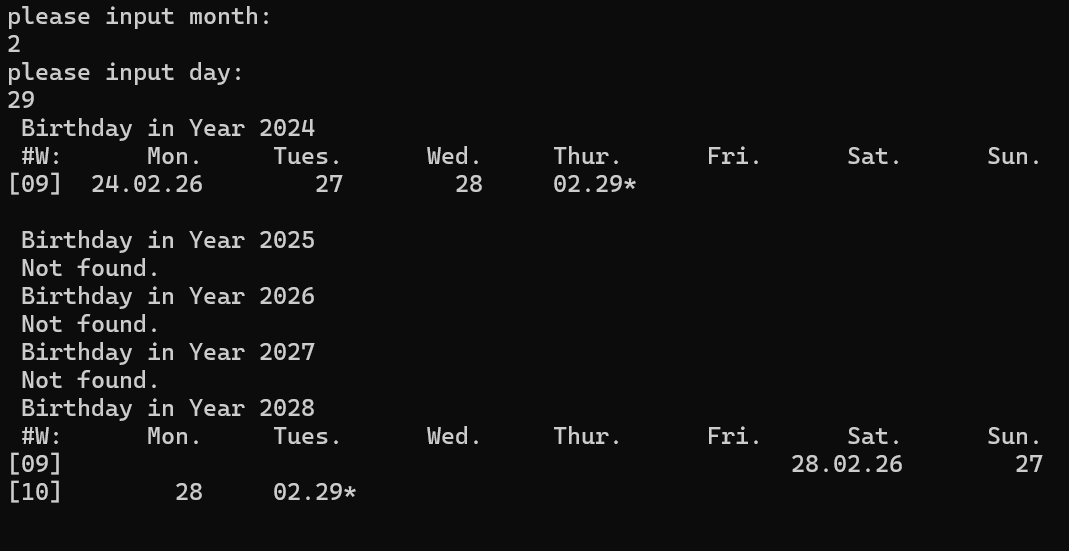


#### 5.3实验测试

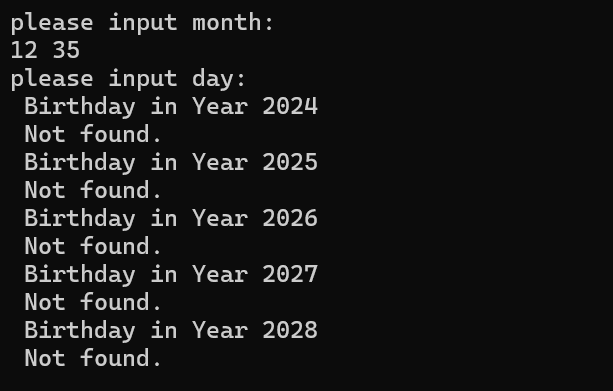
输入1 1



输入 2 29



输入12 35



#### 5.4实验结论

代码测试成功

#### 5.5实验总结

觉得最困难的地方在于处理开年第一周和最后一周，因为有时候判断不了有的日期是算这一年还是前一年或者后一年。

然后结构体函数要改以前很多赋值的部分，有点繁琐

# 本课程学习总结

学习c语言的过程分成两部分吧

第一个部分是学习理论的时候，很多东西记不清，然后一些题就很怪啊QAQ，老是不喜欢写括号啊什么的，顺序很混乱，然后一些运算符运用也不打括号，很不规范。

第二个部分是自己写代码吧，我自己还是觉得写代码感觉还是挺有意思的，写的过程也不会觉得很讨厌，比起线性代数啊什么的好多了，就是改代码的时候很头疼，写了就不想改了，很繁琐。然后vscode去编译多文件一直没用明白，当时想用vscode编多文件程序搞了好久还是没搞懂QAQ，最后还是去用codeblocks了，主要是觉得这个软件太慢了，而且长得有点丑（这是可以说的吗？不可以就当我没有说好吗）。然后用代码解决一些问题的时候也确实比较有成就感，然后自己写代码也不会用很混乱的格式，所以写起来也很谨慎很顺利吧大概，就这样子。

# 附录

完整实验代码附在此处

1. Calendar08实验
2. 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 Comparision 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 ], getWeekSeqOfYear( 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 ;

}

1. Funs.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

1. Funs.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") ;

}

1. 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

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;

}

}

1. Calendar09实验
2. 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 inputMonth, inputDay, Years[YEAR\_NUM];

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;

int i,j;

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

{

printf("\n%1s%s%d\n","","Birthday in Year ", Years[i] );

for( j = 0; j < 366; j ++)

{

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

{

totalNum ++;

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(j == 365)

{

printf(" Not found.\n");

}

}

}

printf("\nTotal %d birthday 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 );

}

}

}

1. Funs.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

1. Funs.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") ;

}

1. 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 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

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 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;

}

}

1. Calendar10实验

main.c #include <stdio.h>

#include <stdlib.h>

#include "date.h"

#include "funs.h"

const int Years[5] = {2024,2025,2026,2027,2028,};

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 inputMonth,inputDay;

int i = 0;

Day birthDay = {0};

Day prepareDay = {0};

Day printDay = {0};

Day prepareday = {0};

// ggb:

printf("please input month: \n");

scanf("%d", &inputMonth) ;

printf("please input day: \n");

scanf("%d", &inputDay) ;

int j = 0;

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

{

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

prepareDay = getDayBefore (birthDay, printDayRange);

prepareday = prepareDay;

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

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

{

printf(" Not found.\n");

continue;

}

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

// printf("#W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

printoneWeek(prepareday);

printf("\n\n");

}

// goto ggb;

return 0 ;

}

1. Funs.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 );

//void printOneDay( Day day);

void printoneWeek( Day day);

#endif // FUNS\_H\_INCLUDED

1. Funs.c

#include <stdio.h>

#include <stdlib.h>

#include "date.h"

#include "funs.h"

extern int Month\_NORMAL\_YEAR [12] ;

extern int Month\_LEAP\_YEAR [12] ;

void printoneWeek( Day prepareday ){

Day remain = prepareday;

int i = 0;

int index = 1;

printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

if(remain.dayseq + 7 - remain.weekDay > isLeapYear(prepareday.year) + 365){

prepareday.weekseq = 1;

}

printf("[%02d]",prepareday.weekseq);

if(i == 0)

{

int j;

for( j = 1; j < remain.weekDay ; j++){

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

}

}

int k;

for( k = 0; k <= printDayRange;k ++){

if(k == 0){

printf(" %02d.%02d.%02d",prepareday.year-100\*(prepareday.year/100), prepareday.month,prepareday.day);

prepareday.day ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}

if(k + remain.weekDay > 7 \*index){

getWeekSeqOfYear( prepareday.year, prepareday.month,getThisSunday( prepareday.year,prepareday.dayseq ));

if(prepareday.dayseq +7 > isLeapYear(prepareday.year) + 365){

prepareday.weekseq = 1;

}

printf("\n");

printf("[%02d]",prepareday.weekseq);

index ++;

}

if(k == printDayRange){

if(remain.dayseq + k > isLeapYear(prepareday.year) + 365){

prepareday.month = 1;

prepareday.day = 1;

prepareday.year ++;

prepareday.dayseq = 1;

prepareday.weekseq = 0;

printf(" %02d.%02d.%02d\*",prepareday.year-100\*(prepareday.year/100) , prepareday.month,prepareday.day);

prepareday.day ++;

prepareday.dayseq ++;

remain.dayseq = 2;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}else if(prepareday.month == 2 && isLeapYear(prepareday.year) == 1 && prepareday.day == 29){

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

prepareday.month++;

prepareday.day = 1;

prepareday.dayseq ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}else if(prepareday.day > Month\_NORMAL\_YEAR[prepareday.month - 1]){

prepareday.day = 1;

prepareday.month ++;

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

prepareday.day++;

prepareday.dayseq ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}else{

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

prepareday.day ++;

prepareday.dayseq ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}

}

if(remain.dayseq + k > isLeapYear(prepareday.year) + 365){

prepareday.month = 1;

prepareday.day = 1;

prepareday.year ++;

prepareday.dayseq = 1;

prepareday.weekseq = 0;

printf(" %02d.%02d.%02d",prepareday.year-100\*(prepareday.year/100) , prepareday.month,prepareday.day);

prepareday.day++;

prepareday.dayseq ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

remain.dayseq = 2;

continue;

}else if(prepareday.month == 2 && isLeapYear(prepareday.year) == 1 && prepareday.day == 29){

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

prepareday.month++;

prepareday.day = 1;

prepareday.dayseq ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}else if(prepareday.day > Month\_NORMAL\_YEAR[prepareday.month - 1]){

prepareday.day = 1;

prepareday.month ++;

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

prepareday.day++;

prepareday.dayseq ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

continue;

}else{

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

prepareday.dayseq += 1;

prepareday.day ++;

prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );

if(prepareday.dayseq > isLeapYear(remain.year) + 365){

prepareday.weekseq = 1;

}

continue;

}

}

}

//void printOneDay( Day prepareday ,int j){

// if(prepareday.weekDay + index > 7){

// break;

// }

// if(prepareday.weekDay > j+1){

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

// }else if(prepareday.weekDay <= j+1){

//

// }

//}

//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") ;

//}

1. 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

#define YEAR\_NUM 5

#define printDayRange 3

typedef struct day{

int year;

int dayseq;

int month;

int day;

int weekseq;

int weekDay;

} Day;

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 );

int getweekDay(int year,int daySeq);

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);

void printDay( Day currentDay, int displayFormat );

#endif // DATE\_H\_INCLUDED

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;

}

}

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

extern int Month\_NORMAL\_YEAR [12] ;

if( isLeapYear(year) != 1 && month == 2 && day >= 29)

{

return -1;

}

if(month < 1 || month > 12){

return -1;

}

if(isLeapYear(year) == 1 && month == 2 && day == 29){

return 1;

}

if(day > Month\_NORMAL\_YEAR [month-1] || day < 1){

return -1;

}

return 1;

}

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

Day ret ;

ret.day = day ;

ret.year = year;

ret.month = month;

ret.dayseq = getDaySeq( year, month, day );

ret.weekseq = getWeekSeqOfYear( year, month, day );

ret.weekDay = getweekDay( year, ret.dayseq);

return ret;

}

Day getDayBefore( Day currentDay,int interval ){

Day ret ;

if(currentDay.month == 1 && currentDay.day - interval <= 0 ){

ret.day = currentDay.day+31 - interval;

ret.month = 12;

ret.dayseq = isLeapYear(currentDay.year -1) + 365 + currentDay.dayseq - interval;

ret.weekDay = currentDay.weekDay - interval <= 0 ? currentDay.weekDay - interval+7 : currentDay.weekDay - interval;

ret.weekseq = getWeekSeqOfYear( currentDay.year - 1, ret.month, ret.day );

ret.year = currentDay.year - 1;

return ret;

}

ret.month = currentDay.day - interval > 0 ?currentDay.month :currentDay.month - 1;

ret.year = currentDay.year;

ret.dayseq = currentDay.dayseq - interval;

ret.day = getDay( ret.year, ret.dayseq );

ret.weekDay = getweekDay(ret.year,ret.dayseq);

ret.weekseq = getWeekSeqOfYear( ret.year, ret.month, ret.day );

return ret;

}

int getweekDay(int year,int daySeq){

int ret = 0;

ret = daySeq+7-getThisSunday(year , daySeq);

return ret ;

}

Day getDayAfter( Day currentDay,int interval );

int getTwoDaysInterval( Day startDay, Day endDay);

void printDay( Day currentDay, int displayFormat );