

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 | 杨清含 | 学号 | U202413716 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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。

# 实验环境

操作系统：Windows 10

编程工具：CodeBlocks 16.01

# 实验一

## 实验任务

* **实验目标**
* 巩固函数的使用，对现有函数进行功能升级以支持全学年校历打印。
* 实现通用功能函数 printOneWeek()，体会“面向过程编程”的设计方法。
* **实验任务**

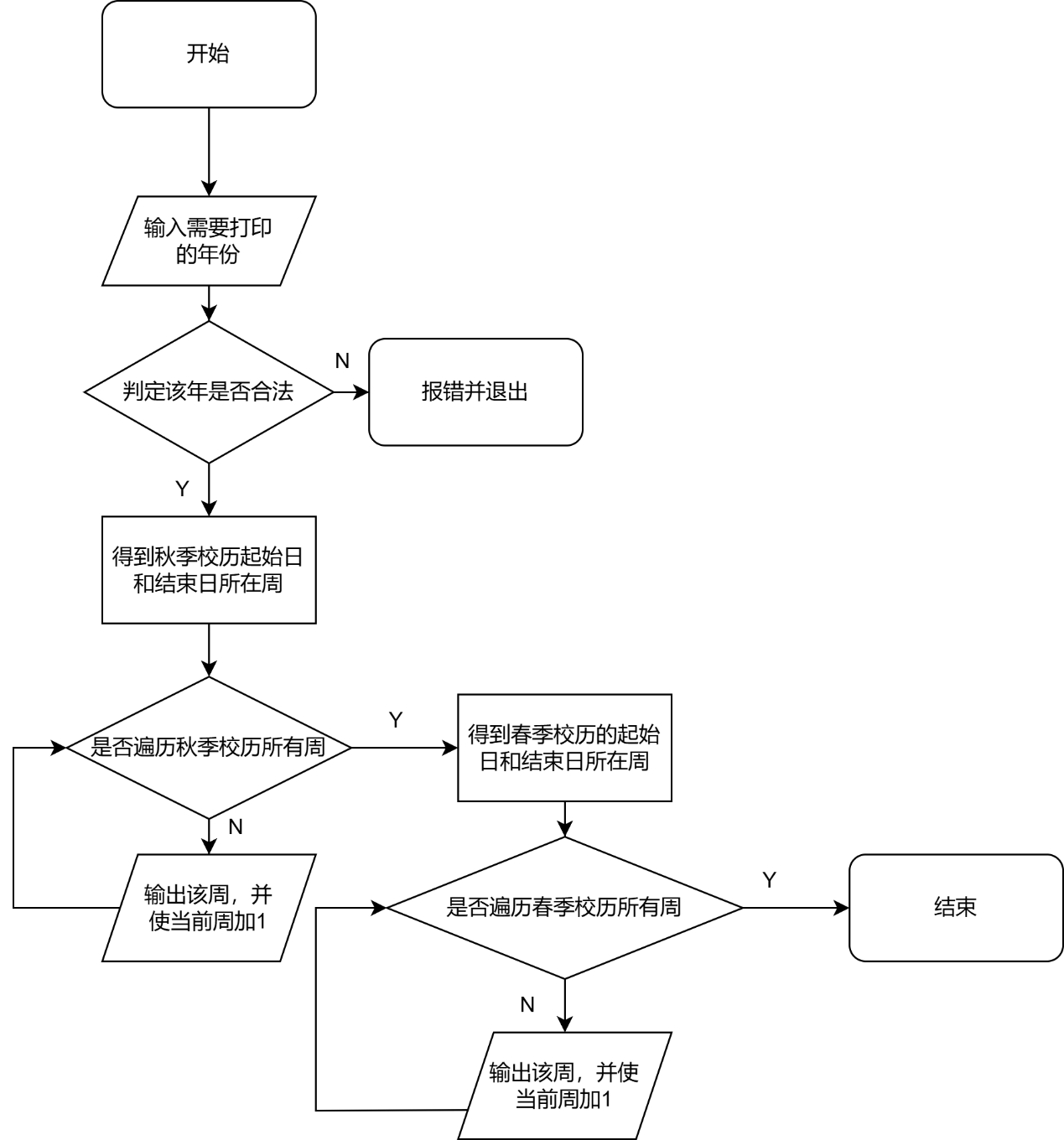
1. **用函数改写打印华中科技大学校历的程序**：打印指定学年度的第一学期和第二学期的校历。
2. **约定学期时间**：
   * 秋季学期：从当年9月4日所在周的周一开始，到第二年1月20日之前一周结束。
   * 春季学期：从次年2月15日之后一周开始，7月第一周结束。
3. **设计打印某周周历的函数**：支持不同学期校历的打印需求。
4. **改进打印某日的函数**：支持在校历首日和跨年日显示年月日信息。

* **函数列表及实现思路**

1. **isLeapYear(int year)**: 判断是否为闰年。
2. **getDaySeqOnJan1(int year)**: 获取1月1日是一年中的第几天。
3. **getMonthLength(int year, int month)**: 获取指定月份的天数。
4. **getDaySeq(int year, int month, int day)**: 获取指定日期是一年中的第几天。
5. **getWeekSeqOfYear(int year, int month, int day)**: 获取指定日期是一年中的第几周。
6. **getMonth(int year, int daySeqOfYear)**: 根据一年中的第几天获取月份。
7. **getDay(int year, int daySeqOfYear)**: 根据一年中的第几天获取日期。
8. **getDaySeqOfWeek(int year, int daySeqOfYear)**: 获取指定日期是所在周的第几天。
9. **getNextMonday(int year, int daySeqOfYear)**: 获取指定日期后的第一个星期一的一年中的第几天。
10. **getThisMonday(int year, int daySeqOfYear)**: 获取指定日期所在周的星期一的一年中的第几天。
11. **getThisSunday(int year, int daySeqOfYear)**: 获取指定日期所在周的星期日的一年中的第几天

## 实验步骤

编程思路



获得某日所在周的函数

1. **int** getWeekOfYear(**int** year, **int** month, **int** day)
2. {
3. **int** daySeqOfYear = getDaySeq(year, month, day);
4. daySeqOfYear = daySeqOfYear + getDaySeqOnJan1(year) - 1;
5. **int** week = daySeqOfYear / 7 + 1;
6. **return** week;
7. }

打印一整周的函数

1. **void** printOneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow)
2. {
3. // Before Monday, print the week sequence
4. printf("[%02d]", weekSeqShow);
6. // Declare variables to navigate the semester
7. **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - getDaySeqOnJan1(year) + 2;
8. **int** sEndSeqOfYear = sStartSeqOfYear + 6;
9. **int** currentYearLength = isLeapYear(year);
10. **int** daySeqOfYear = sStartSeqOfYear;
12. // If this is the first week of semester, show full month and day for first day
13. **if** ( weekSeqShow == 1 )
14. {
15. printOneDay( year, daySeqOfYear, DATE\_INFO\_FULL );
16. daySeqOfYear ++;
17. }
19. // Print the semester calendar
20. **for** ( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )
21. {
22. // Print the month and day for the current day
23. **if** ( daySeqOfYear <= currentYearLength )
24. {
25. printOneDay( year, daySeqOfYear, DATE\_INFO\_BRIEF );
26. }
27. **else** **if**(daySeqOfYear == currentYearLength + 1)
28. {
29. printOneDay(year + 1, daySeqOfYear - currentYearLength, DATE\_INFO\_FULL);
30. }
31. **else**
32. {
33. printOneDay(year + 1, daySeqOfYear - currentYearLength, DATE\_INFO\_BRIEF);
34. }
35. }
37. // After Sunday, print a new line
38. printf("\n");
39. }

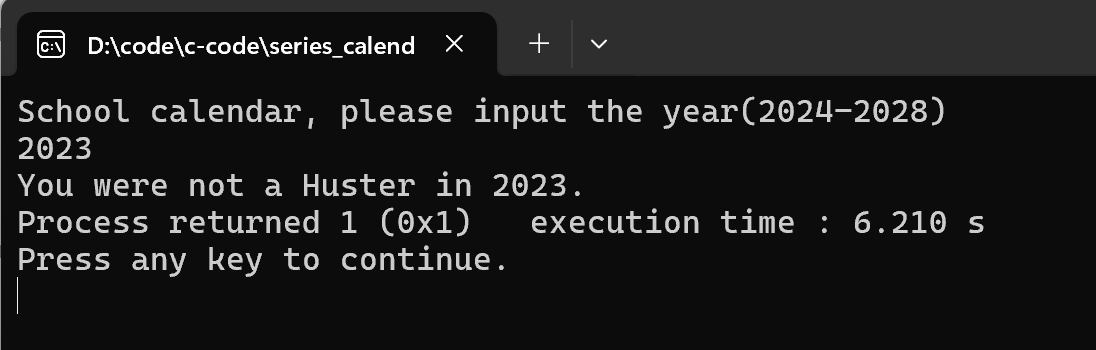
## 代码测试

### 测试该程序对异常年能否正常报错

测试步骤：输入小于YEARMIN的年份，检测程序能否正常报错并退出

预计测试结果：输出You were not a Huster in %d.并退出

实测结果：



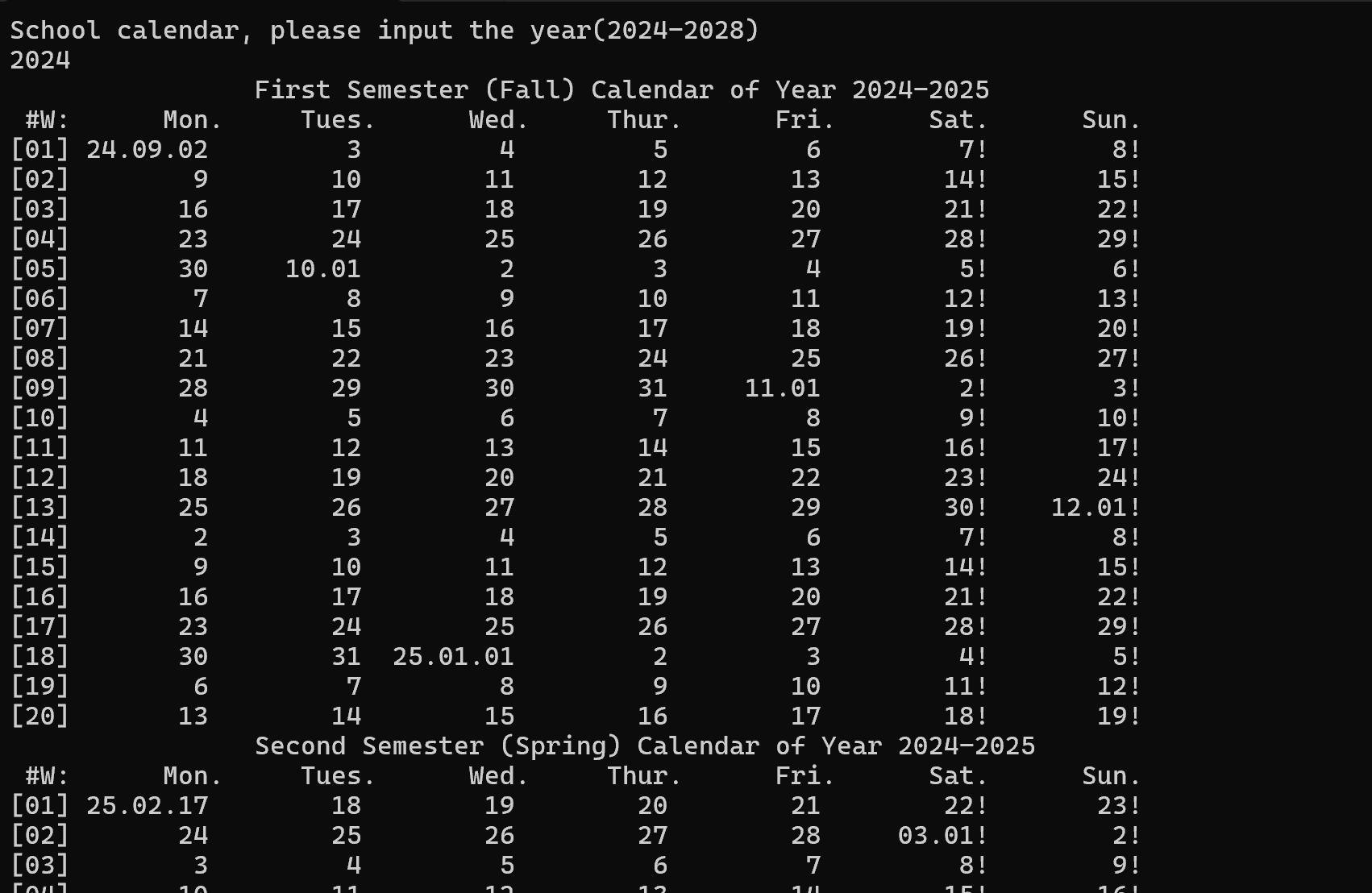
测试结论：该测试点通过

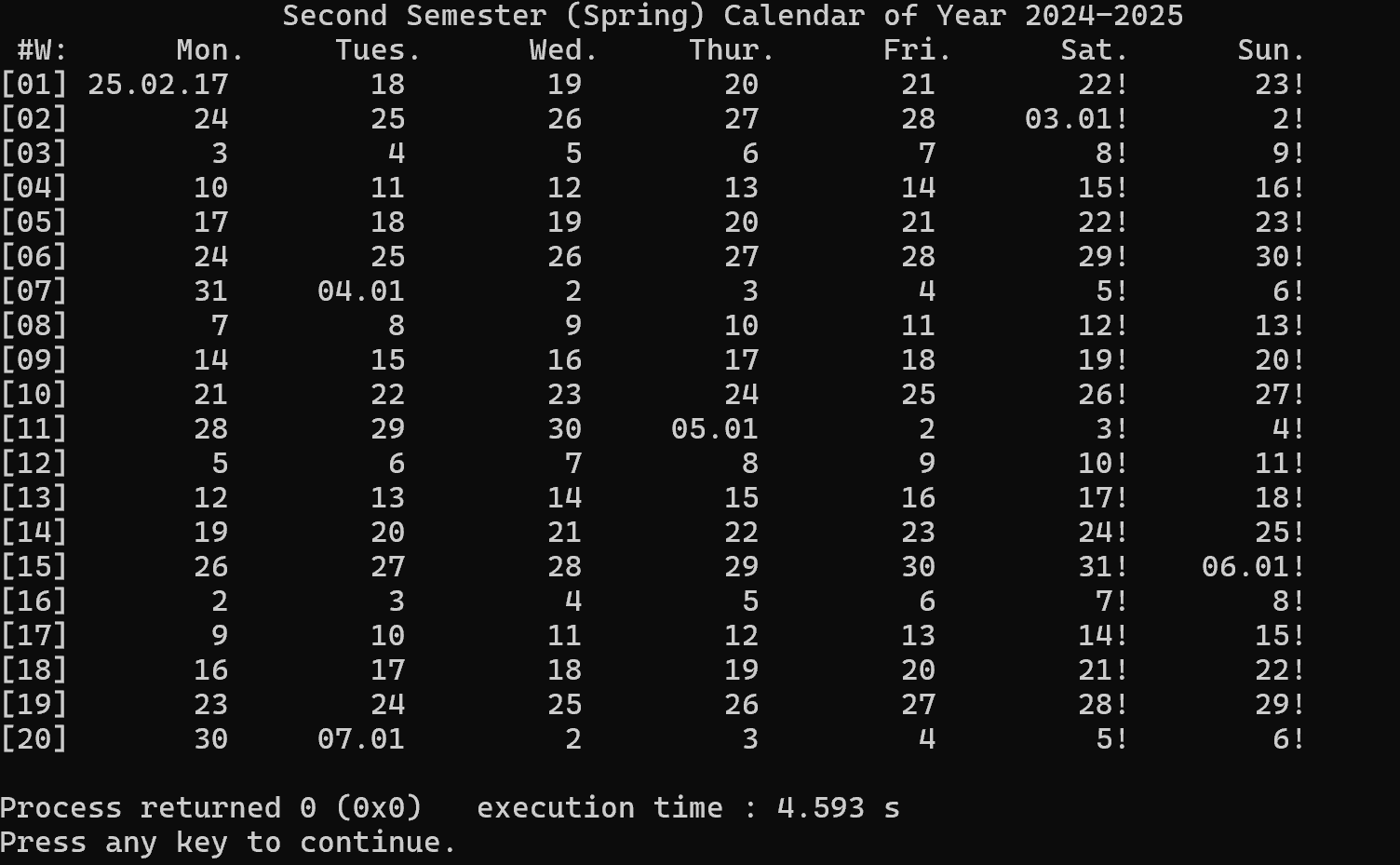
### 测试该程序能否正常输出校历

测试步骤：输入2024-2028年中的一年，检测程序能否正常输出校历

预计测试结果：输出该年的正常校历

实测结果：





测试结果：该程序点通过

## 实验结论

代码能够达到功能目标

## 实验总结

本实验运用自顶而下逐步求精的编程思想，先通过main函数搭建好框架，后通过运用函数及程序控制语句对数据运算过程进行求精，实现了面向过程的编程，同时体会了多文件工程的操作及其用法。

但仍存在以下问题：

* 1. 由于过程本身的复杂性，在date计算类的函数表现出逻辑的复杂和可读性较低
  2. 在过程计算时，没有达到最优解
  3. 由于没有掌握多工程编程的思路，没有为每一个函数建立测试程序，导致程序在出现逻辑错误后，排除bug难度较大，影响编程进度

# 实验二

## 实验任务

* **实验目标**
* 学习多维数组的使用。
* 利用多维数组事先存储所需处理的数据。
* 体会“过程中生成数据”与“生成数据再操作”两种编程思路的差异。
* **实验任务**

1. **用三维数组记录多年的日期数据**：定义一个三维数组来存储多年的日期信息。
2. **查找某人的生日**：输入某人的生日，查找其在数组中的记录。
3. **打印该生日所在周的周历**：打印出该生日所在周的详细信息。
4. **计算其在周末过生日的次数**：计算该人在周末过生日的次数。

* **三维数组的定义**

c复制

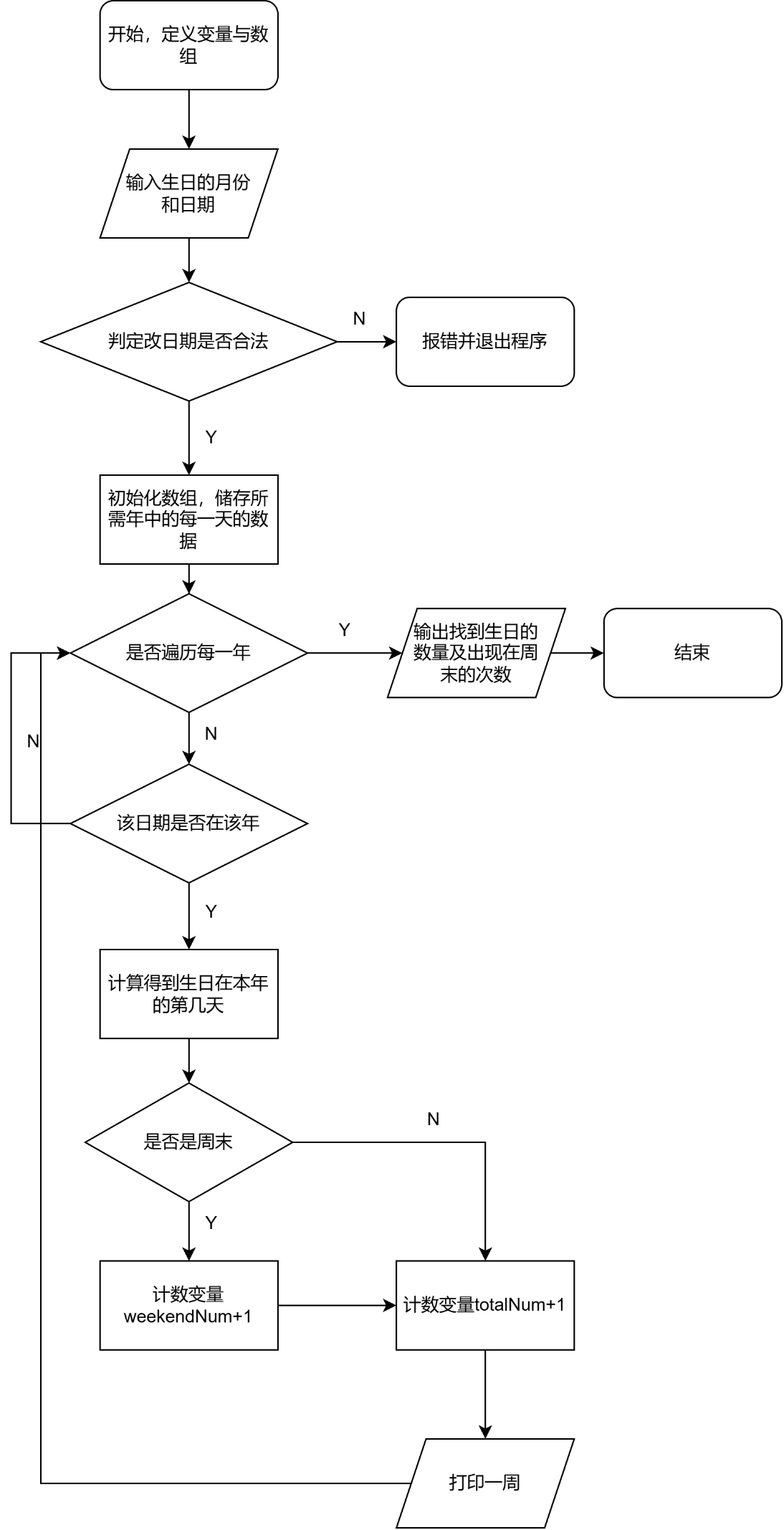
int Days[YEAR\_NUM][366][5] = {0};

* 第一维：年份，例如 2018, 2019, 2020, ...
* 第二维：一年中的第几天，例如 1, 2, 3, ..., 365, 366
* 第三维：每天的详细信息
  + Days[..][..][0]: 月份
  + Days[..][..][1]: 日期
  + Days[..][..][2]: 一年中的第几周
  + Days[..][..][3]: 一周中的第几天
* **函数列表及实现思路**

1. **初始化数组**：填充三维数组，记录每年的日期信息。
2. **查找生日**：根据用户输入的生日查找对应的记录。
3. **打印周历**：打印指定周的详细信息。
4. **计算周末过生日的次数**：计算用户在周末过生日的次数。

## 实验步骤

编程思路



初始化年份与日期的函数

1. **void** initialDays(**int** Years[], **int** Days[][366][4], **int** yearNum)
2. {
3. **int** year, month, day, yearLength, weekSeq, seqofweek;
4. **int** i, j;
6. **for** (i = 0; i < yearNum; i++)
7. {
8. year = Years[i];
9. yearLength = isLeapYear(year);
10. **for** (j = 0; j < yearLength; j++)
11. {
12. Days[i][j][0] = getMonth(year, j + 1);
13. Days[i][j][1] = getDay(year, j + 1);
14. Days[i][j][2] = getWeekSeqOfYear(year, Days[i][j][0], Days[i][j][1]);
15. Days[i][j][3] = getDaySeqOfWeek(year, j + 1);
16. }
17. }
19. **return**;
20. }
22. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin)
23. {
24. **int** i = 0;
25. **for**(i = 0; i < yearNum; i++)
26. {
27. Years[i] = yearMin + i;
28. }
30. **return** 0;
31. }

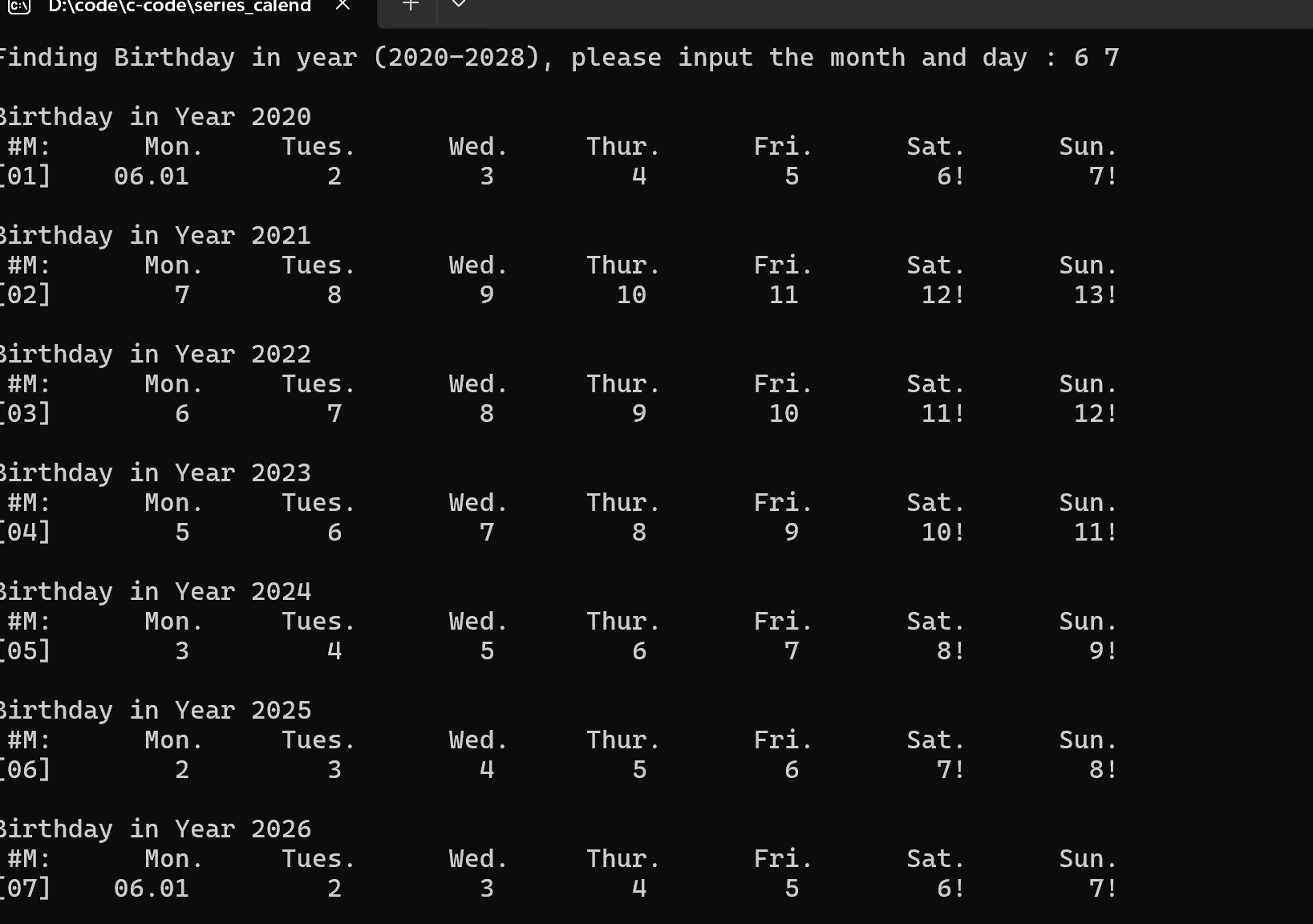
## 代码测试

### 测试代码能否正常计数普通日期的生日

测试步骤：输入普通日期（非2.29）测试程序能否正常打印并计数周末数与生日数

预计结果：程序正常输出，并计数正常

实测结果：



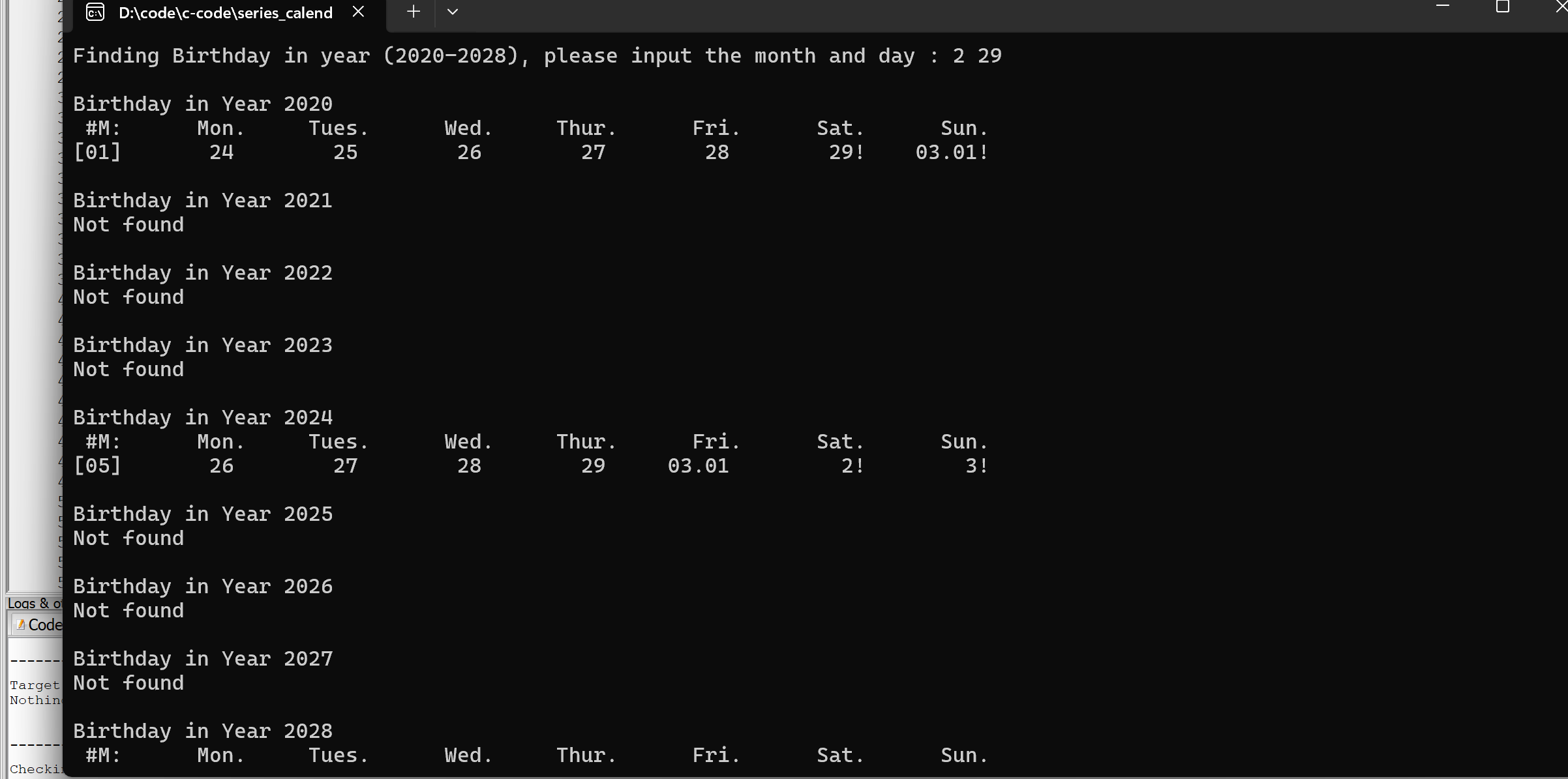
实验结论：测试通过

### 测试代码能否打印2 29并正常输出not found

测试步骤：输入2 29并观察终端打印结果

预计结果：在非闰年打印not found在闰年打印该周

实测结果：

****

测试结论：测试通过

## 实验结论

该程序能完成目标任务

## 实验总结

实验通过数组对一年中的各种信息进行储存，使我更清楚的理解内存与运行的关系

仍存在以下问题：

1：在跨年时出现逻辑错误，导致打印周历时出现负数

2：未充分利用内存，导致在有充足的内存后依然采用复杂的过程

# 实验三

## 实验任务

* **实验目标**
* 学习如何使用结构体来封装日期数据。
* 理解“面向过程的编程”与“面向数据类型的编程”在设计上的差异。
* 实现日期的赋值和偏移天数等操作。
* **实验任务**

1. **输入某人的生日**：通过用户输入获取生日日期。
2. **日期偏移计算**：计算生日前三天的日期。
3. **打印相关周历**：显示这三天的日期信息。

* **结构体定义**

typedef struct {

int year; *// 年份*

int month; *// 月份*

int day; *// 日期*

int daySeq; *// 一年中的第几天*

int weekSeq; *// 一年中的第几周*

int weekDay; *// 一周中的第几天*

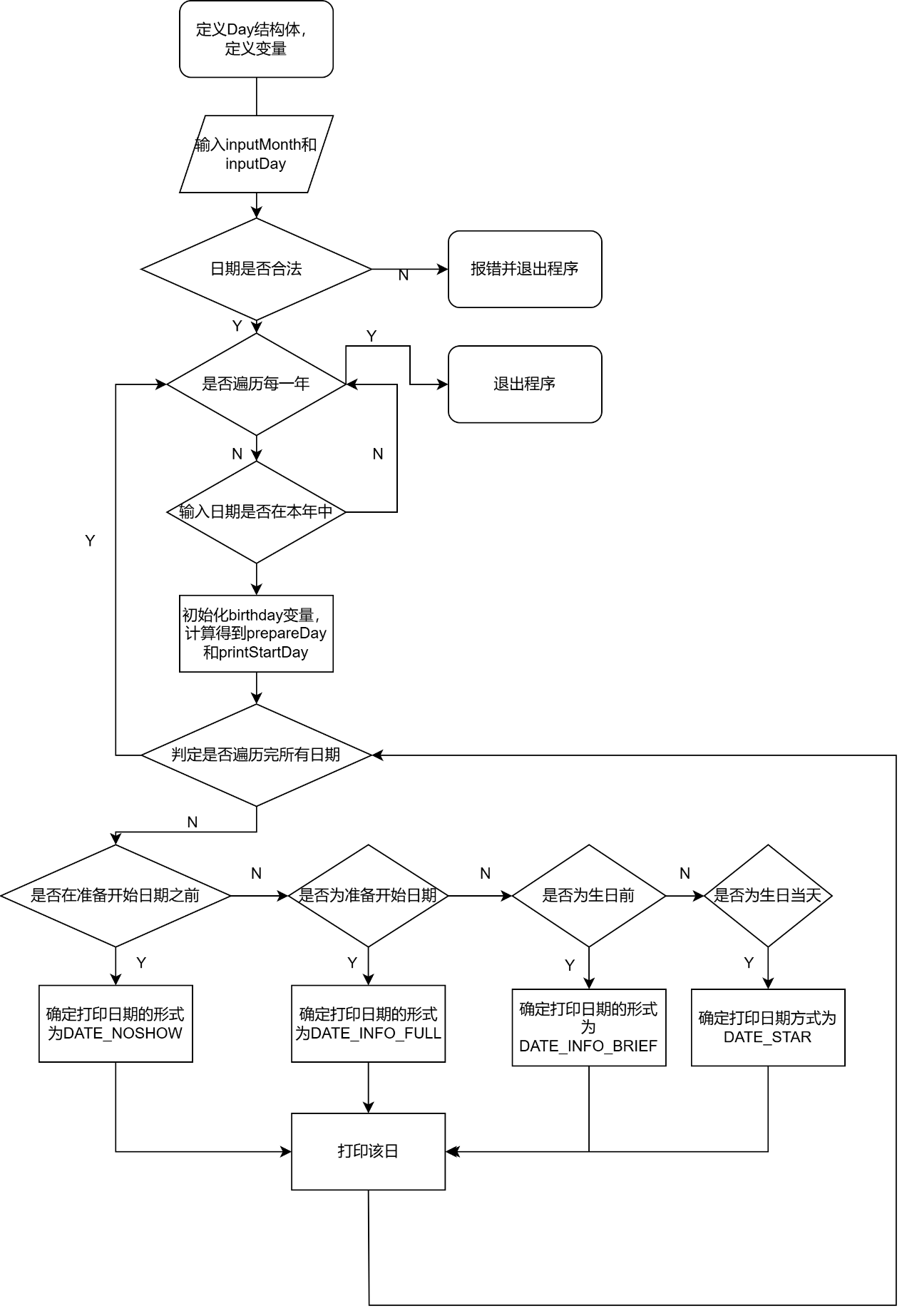
} Day;

* **函数列表及实现思路**

1. **isLeapYear(int year)**: 判断是否为闰年。
2. **getDaySeqOnJan1(int year)**: 获取1月1日是一年中的第几天。
3. **getDaySeq(int year, int month, int day)**: 获取指定日期是一年中的第几天。
4. **isDay(int year, int month, int day)**: 检查日期是否有效。
5. **setDay(int year, int month, int day)**: 设置日期并返回Day结构体。
6. **getDayBefore(Day currentDay, int interval)** 和 **getDayAfter(Day currentDay, int interval)**: 计算给定日期前后的日期。
7. **printDay(Day currentDay, int displayFormat)**: 打印日期。

## 实验步骤

编程思路



关键代码：

初始化日期的函数

1. Day setDay(**int** year, **int** month, **int** day)
2. {
3. Day d;
4. d.year = year;
5. d.month = month;
6. d.day = day;
7. d.daySeq = getDaySeq(year, month, day);
8. d.weekSeq = getWeekSeqOfYear(year, month, day);
9. d.weekDay = getDaySeqOfWeek(year, d.daySeq);
11. **return** d;
12. }

调整日期的代码

1. Day getDayBefore(Day currentday, **int** interval)
2. {
3. currentday.daySeq -= interval;
4. **while**(currentday.daySeq <= 0)
5. {
6. currentday.year --;
7. currentday.daySeq += isLeapYear(currentday.year );
8. }
10. currentday.month = getMonth(currentday.year, currentday.daySeq);
11. currentday.day = getDay(currentday.year,currentday.daySeq);
12. currentday.weekDay = getDaySeqOfWeek(currentday.year, currentday.daySeq);
13. currentday.weekSeq = getWeekSeqOfYear(currentday.year, currentday.month, currentday.day);
15. **return** currentday;
16. }

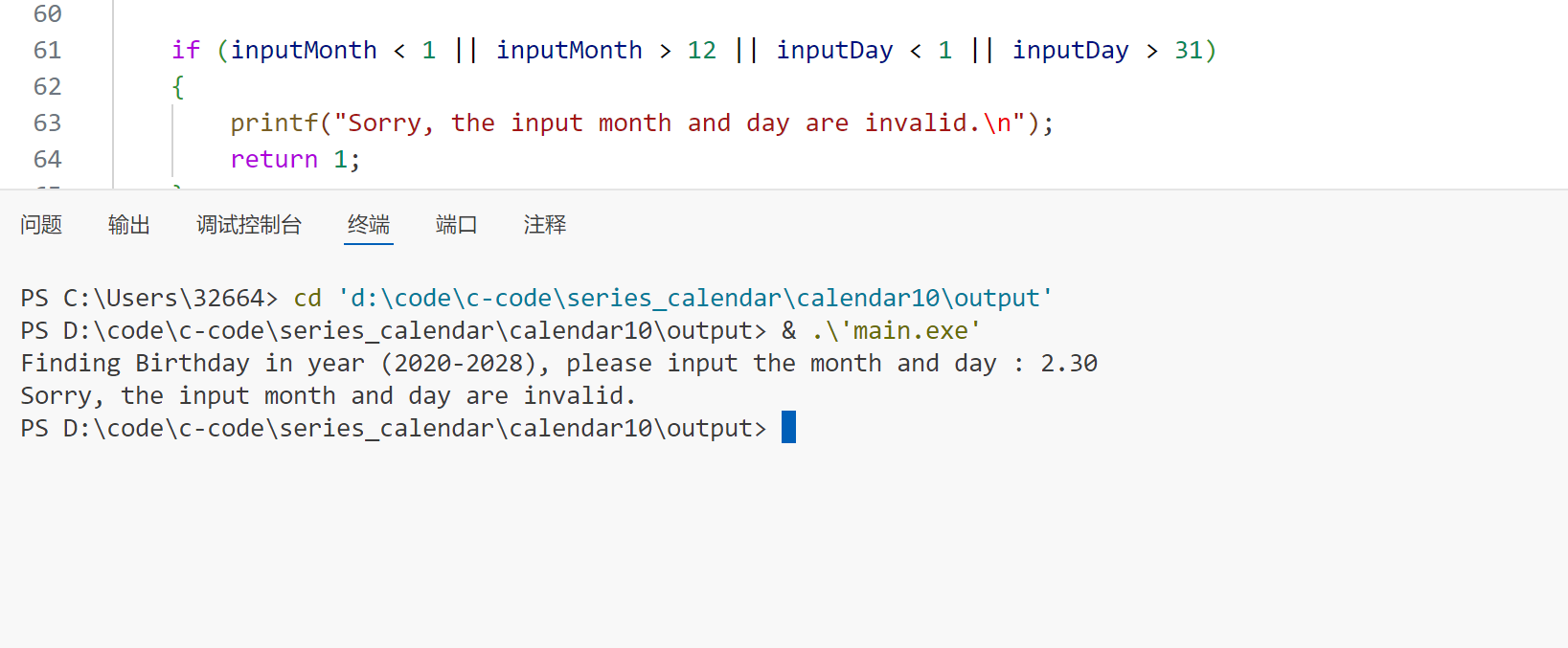
## 代码测试

### 测试判定日期是否合法的程序的测试结果

测试步骤：输入非法日期（如2.30），检测程序能否正常报错

预期程序测试结果：程序退出，输出Sorry, the input month and day are invalid.

实测结果



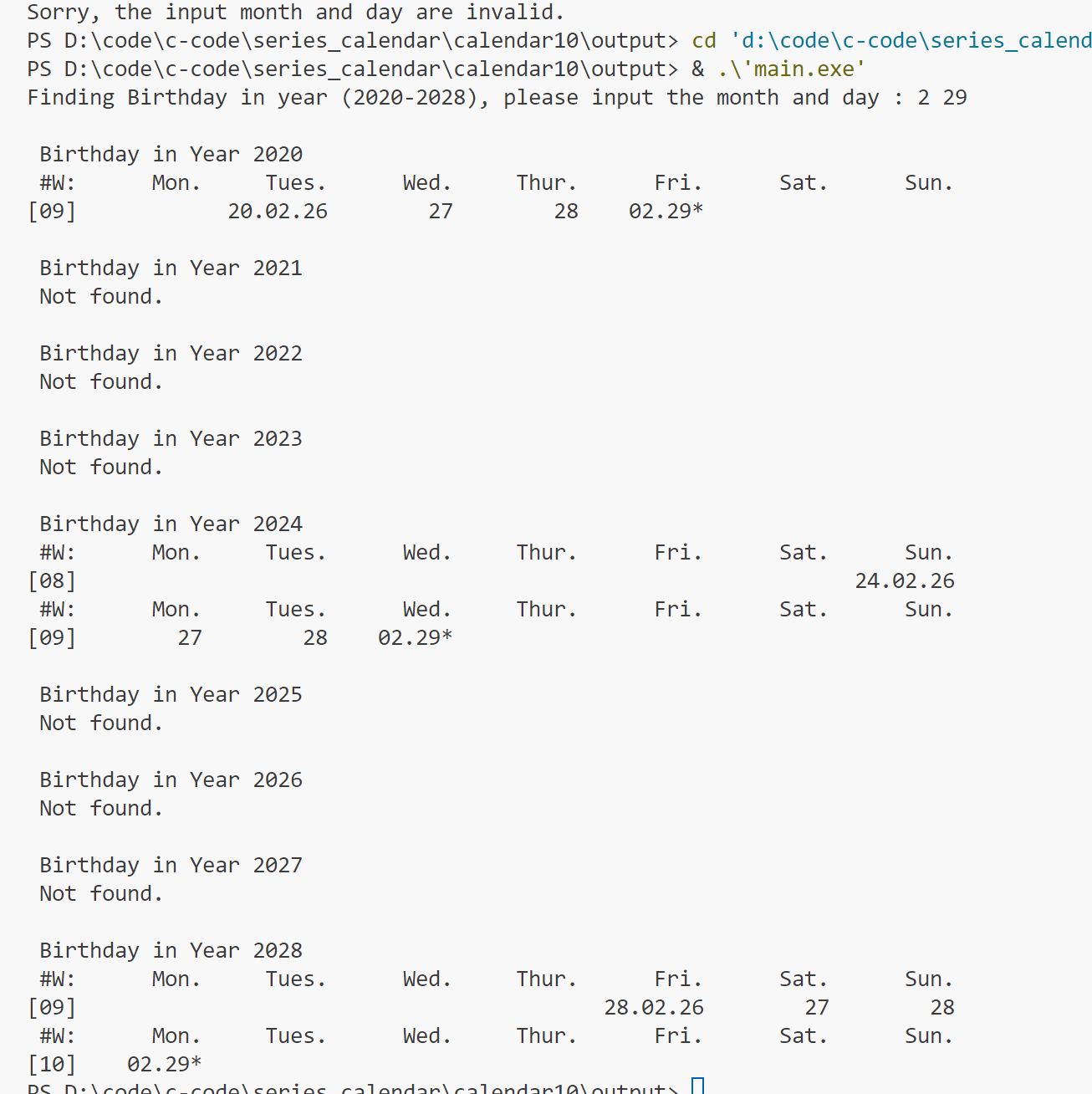
测试结论：该测试点通过

### 测试程序对于2.29的特殊日期正常输出

测试步骤：输入2.29，检测程序是否能正常输出

预期测试结果：在非闰年输出 Not found.在闰年输出正常

实测结果：



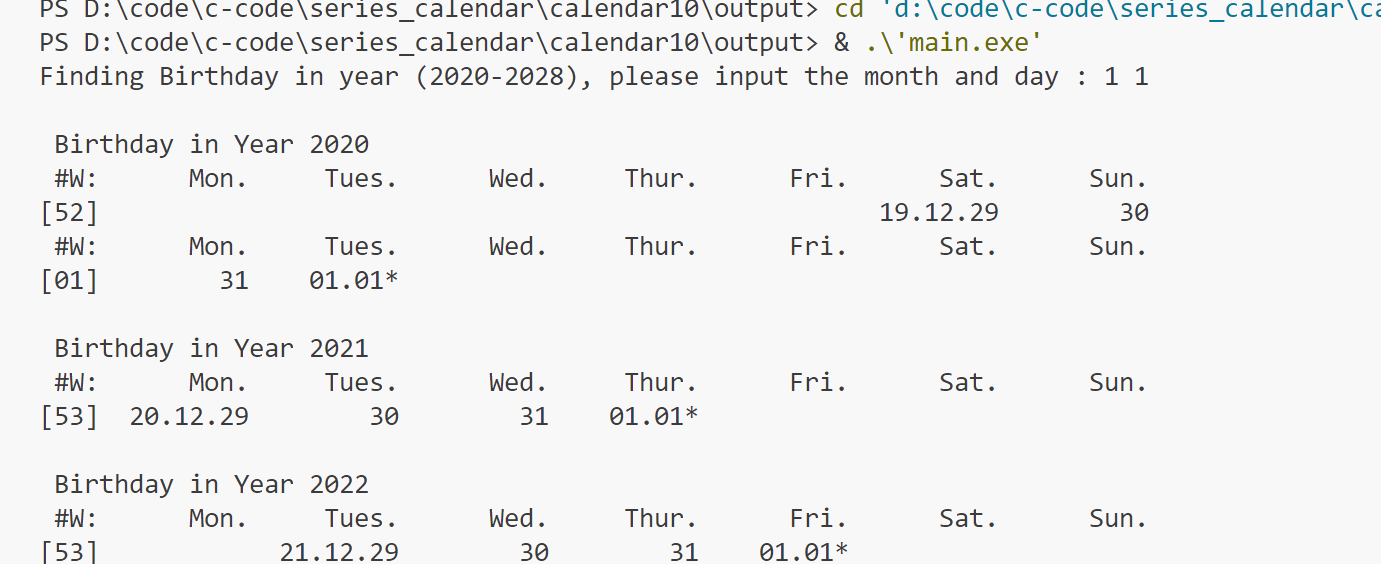
测试结论：测试点通过

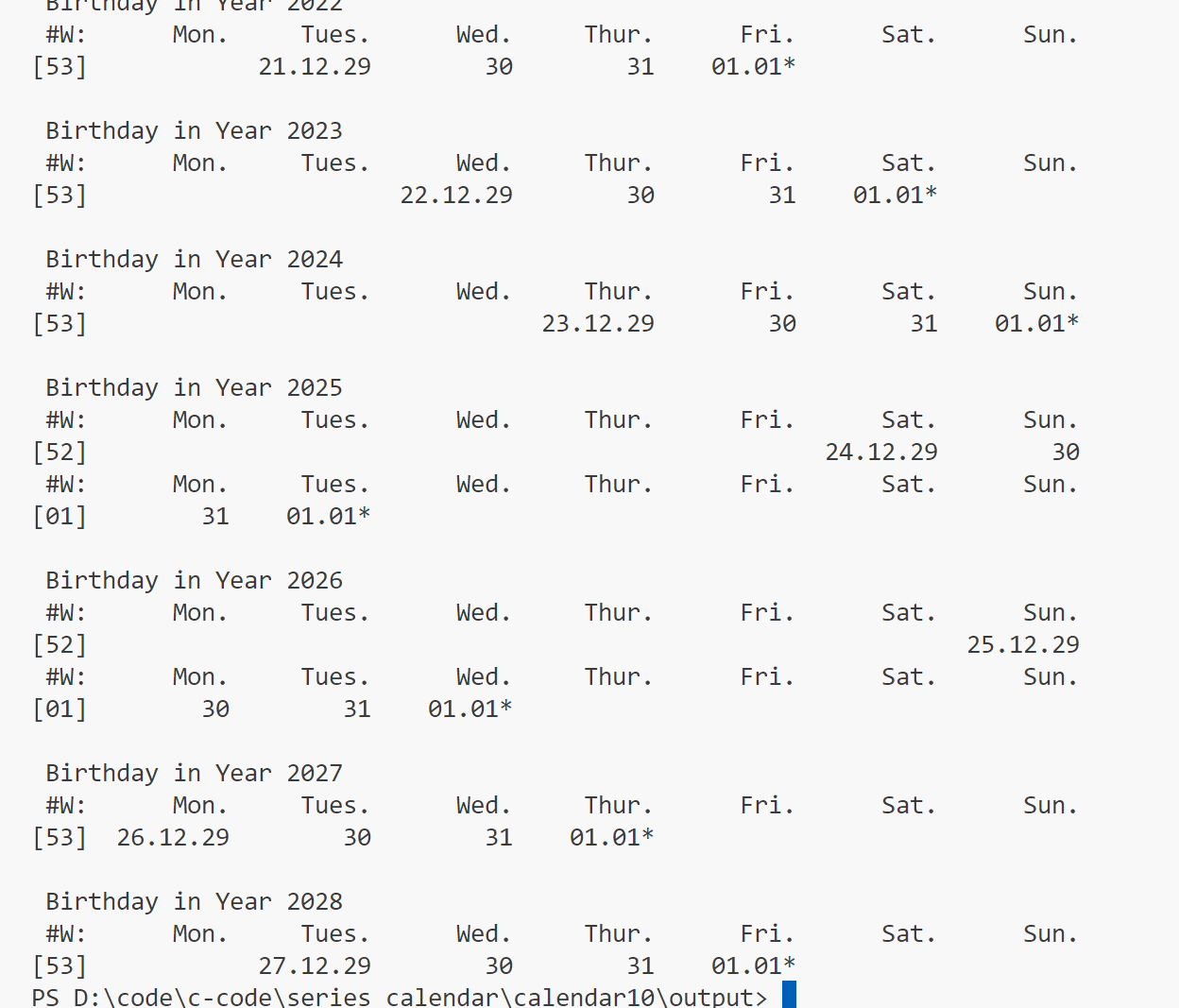
### 测试程序能否正常处理跨年日期

测试步骤：输入1 1，检测程序能否正常输出

预期结果：在准备周时打印周数为52或53，在元旦周打印周数1

实测结果：





## 实验结论

该程序能正常完成功能

## 实验总结

实验按照自顶而下逐步求精的编程思想进行，先通过main函数搭好面向对象的编程的逻辑框架，之后通过对函数的进一步搭建，实现了对结构体的过程操作

但仍存在如下问题：

* 1. 代码不够精简，冗余过多
  2. 对常数组MONTH\_NORMAL\_YEAR等的利用不到位，补充了很多冗余的函数如getMonthLength等
  3. 逻辑处理不完善，isLeapYear不是通常意义的输出bool值0或1，二十直接输出该年的天数，降低了代码的可读性

# 本课程学习总结

**初识C语言**

* **入门阶段**：开始学习C语言时，主要是理解基本的语法和结构，如变量、数据类型、运算符、控制结构（if语句、for循环等）。
* **函数和数组**：随着学习的深入，开始接触函数和数组，这是C语言编程中非常重要的概念，用于代码的模块化和数据的组织。
* **指针和内存管理**：指针是C语言中一个高级且强大的特性，它允许直接操作内存地址，但同时也是初学者容易出错的地方。

**进阶学习**

* **结构体和多维数组**：学习如何使用结构体来组织复杂的数据，以及多维数组在处理大量数据时的应用。

**程序中出现的问题和改进**

**常见问题**

* **语法错误**：初学者常犯的错误，如遗漏分号、括号不匹配等。
* **逻辑错误**：程序的逻辑不符合预期，如循环条件错误、数组越界等。
* **内存泄漏**：由于不正确的内存分配和释放导致的问题。

**改进方法**

* **代码审查**：定期审查自己的代码，检查潜在的错误和改进点。
* **使用编译器警告**：利用编译器的警告功能来发现潜在的问题。
* **单元测试**：编写测试用例来验证代码的正确性，特别是在处理复杂逻辑时。
* **代码规范与调试技巧**

**代码规范**

* **命名规范**：使用有意义的变量名和函数名，避免使用缩写。
* **代码格式化**：保持一致的缩进和空格使用，使代码更易读。
* **注释**：在代码中添加必要的注释，解释复杂的逻辑和设计决策。

**调试技巧**

* **使用调试器**：利用调试器逐步执行程序，观察变量的变化和程序的执行流程。
* **打印调试信息**：在关键位置添加打印语句，输出调试信息。
* **简化问题**：将复杂的问题分解为更小的部分，逐一解决。
* **学习记录和心得**

**学习记录**

* **笔记**：记录学习过程中的重要概念、示例代码和个人理解。
* **项目实践**：通过实际项目来应用所学知识，如编写一个简单的计算器程序或数据处理工具。
* **在线课程和书籍**：利用在线资源和书籍来补充和深化理解。

**心得体会**

* **耐心和坚持**：学习编程需要时间和耐心，不断练习和尝试是提高的关键。
* **解决问题的能力**：编程不仅是写代码，更重要的是解决问题的能力，这需要不断学习和实践。
* **持续学习**：技术在不断进步，持续学习新技术和工具是非常重要的。

总的来说，学习C语言是一个不断探索和挑战自我的过程。通过不断的实践、反思和学习，可以逐渐提高编程技能，解决更复杂的问题，并在编程的道路上不断前进。

# 附录

## 实验3代码

### Main.c下的代码

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include"date.h"
4. #include"fun.h"
6. **int** main()
7. {
8. **int** inputYear;
10. printf("School calendar, please input the year(%d-%d)\n", YEAR\_MIN,YEAR\_MAX);
11. scanf("%d", &inputYear);
13. **if**(inputYear < YEAR\_MIN)
14. {
15. printf("You were not a Huster in %d.", inputYear);
16. **return** 1;
17. }
18. **else** **if** (inputYear > YEAR\_MAX)
19. {
20. printf("Congratulations! You'll have graduated in %d.",inputYear);
21. **return** 2;
22. }
24. printf("\t\tFirst Semester (Fall) Calendar of Year %d-%d\n", inputYear, inputYear + 1);
25. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n","Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.");
27. **int** startWeek = getWeekOfYear(inputYear, 9, 4);
28. **int** endWeek = getWeekOfYear(inputYear, 12, 25) + getWeekOfYear(inputYear + 1, 1, 20) - 1;
29. **int** weekSeqOfYear = startWeek;
30. **int** showWeek = 1;
32. **for**(; weekSeqOfYear <= endWeek; weekSeqOfYear ++, showWeek++)
33. {
34. printOneWeek(inputYear, weekSeqOfYear, showWeek);
35. }
37. printf("\t\tSecond Semester (Spring) Calendar of Year %d-%d\n", inputYear, inputYear + 1);
38. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n","Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.");
40. startWeek = getWeekOfYear(inputYear + 1, 2, 22);
41. endWeek = getWeekOfYear(inputYear + 1, 7, 1);
42. weekSeqOfYear = startWeek;
43. showWeek = 1;
45. **for**(; weekSeqOfYear <= endWeek; weekSeqOfYear ++, showWeek++)
46. {
47. printOneWeek(inputYear + 1, weekSeqOfYear, showWeek);
48. }
49. **return** 0;
50. }

### 头文件的代码

Fun.h

1. #ifndef FUN\_H\_INCLUDED
2. #define FUN\_H\_INCLUDED


6. #endif // FUN\_H\_INCLUDED
8. #define YEAR\_MIN           2024
9. #define YEAR\_MAX           2028
10. #define DATE\_INFO\_BRIEF    1
11. #define DATE\_INFO\_FULL     2
13. **void** printOneDay(**int** year, **int** daySeqOfYear,**int** formatType);
14. **void** printOneWeek(**int** year, **int** weekSeqOfYear,**int** weekSeqShow);

date.h

1. #ifndef DATE\_H\_INCLUDED
2. #define DATE\_H\_INCLUDED


6. #endif // DATE\_H\_INCLUDED
7. **int** isLeapYear(**int** year);
8. **int** getDaySeqOnJan1(**int** year);
10. // Functions about month and day
11. **int** getMonthLength(**int** year, **int** month);
12. **int** getDaySeq(**int** year, **int** month, **int** day);
13. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day);
15. // Functions for properties of one day
16. **int** getMonth(**int** year, **int** daySeqOfYear);
17. **int** getDay(**int** year, **int** daySeqOfYear);
18. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear);
20. // Functions for day movement calculation
21. **int** getNextMonday(**int** year, **int** daySeqOfYear);
22. **int** getThisMonday(**int** year, **int** daySeqOfYear);
23. **int** getThisSunday(**int** year, **int** daySeqOfYear);

### 非main函数的c文件

date.c

1. #include "date.h"
2. #include "fun.h"
3. **const** **int** Month\_NORMAL\_YEAR[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
4. **const** **int** Month\_LEAP\_YEAR[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
5. **int** isLeapYear(**int** year)
6. {
7. **if** (year % 4 == 0 && year % 100 != 0)
8. {
9. **return** 366;
10. }
11. **else**
12. {
13. **return** 365;
14. }
15. }
16. **int** getDaySeqOnJan1(**int** year)
17. {
18. **int** day = 1;
19. **while** (year > 2024)
20. {
21. day += isLeapYear(year - 1);
22. year --;
23. }
24. **while**(year < 2024)
25. {
26. day -= isLeapYear(year);
27. year ++;
28. }
30. **if** (day > 0)
31. {
32. day %= 7;
33. }
34. **else**
35. {
36. day = day % 7 + 7;
37. }
38. **return** day;
39. }
41. **int** getMonthLength(**int** year, **int** month)
42. {
43. **int** day;
44. **switch**(month)
45. {
46. **case** 1:
47. day = 31;
48. **break**;
49. **case** 2:
50. **if** (year % 4 == 0 && year % 100 != 0)
51. {
52. day = 29;
53. }
54. **else**
55. {
56. day = 28;
57. }
58. **break**;
59. **case** 3:
60. day = 31;
61. **break**;
62. **case** 4:
63. day = 30;
64. **break**;
65. **case** 5:
66. day = 31;
67. **break**;
68. **case** 6:
69. day = 30;
70. **break**;
71. **case** 7:
72. day = 31;
73. **break**;
74. **case** 8:
75. day = 31;
76. **break**;
77. **case** 9 :
78. day = 30;
79. **break**;
80. **case** 10:
81. day = 31;
82. **break**;
83. **case** 11:
84. day = 30;
85. **break**;
86. **case** 12:
87. day = 31;
88. **break**;
89. }
91. **return** day;
92. }
93. **int** getDaySeq(**int** year, **int** month, **int** day)
94. {
95. /\*for(;month > 1 ;month --)
96. {
97. day += getMonthLength(year, month);
98. }\*/
99. **if**(isLeapYear(year) == 365)
100. {
101. **for**(;month > 1; month --)
102. {
103. day += Month\_NORMAL\_YEAR[month - 2];
104. }
105. }
106. **else**
107. {
108. **for**(;month > 1; month --)
109. {
110. day += Month\_LEAP\_YEAR[month - 2];
111. }
112. }
113. **return** day;
114. }
115. **int** getWeekOfYear(**int** year, **int** month, **int** day)
116. {
117. **int** daySeqOfYear = getDaySeq(year, month, day);
118. daySeqOfYear = daySeqOfYear + getDaySeqOnJan1(year) - 1;
119. **int** week = daySeqOfYear / 7 + 1;
120. **return** week;
121. }
123. **int** getMonth(**int** year, **int** daySeqOfYear)
124. {
125. **int** month ;
126. **for**(month = 1; daySeqOfYear > getMonthLength(year, month); month ++)
127. {
128. daySeqOfYear -= getMonthLength(year, month);
129. }
130. **return** month;
131. }
132. **int** getDay(**int** year, **int** daySeqOfYear)
133. {
134. **int** month ;
135. **for**(month = 1;daySeqOfYear > getMonthLength(year, month); month ++)
136. {
137. daySeqOfYear -= getMonthLength(year, month);
138. }
139. **return** daySeqOfYear;
140. }
141. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
142. {
143. **int** day = (daySeqOfYear + getDaySeqOnJan1(year) - 1) % 7;
144. **return** day;
145. }
147. **int** getNextMonday(**int** year, **int** daySeqOfYear)
148. {
149. **int** day = (getDaySeqOfWeek(year, daySeqOfYear) == 1) ?
150. daySeqOfYear : daySeqOfYear + 8 - getDaySeqOfWeek(year, daySeqOfYear);
151. **return** day;
152. }
153. **int** getThisMonday(**int** year, **int** daySeqOfYear)
154. {
155. **int** day = daySeqOfYear - getDaySeqOfWeek(year, daySeqOfYear) + 1;
156. **return** day;
157. }
158. **int** getThisSunday(**int** year, **int** daySeqOfYear)
159. {
160. **int** day = (getDaySeqOfWeek(year,daySeqOfYear) == 0) ?
161. daySeqOfYear : daySeqOfYear + 7 - getDaySeqOfWeek(year,daySeqOfYear);
163. **return** day;
164. }

### fun.c

1. #include "fun.h"
2. #include "date.h"
4. **void** printOneDay(**int** year, **int** daySeqOfYear,**int** formatType)
5. {
6. **int** month = getMonth(year, daySeqOfYear);
7. **int** day = getDay(year, daySeqOfYear);
9. **if**(formatType == 2 )
10. {
11. printf(" %02d.%02d.%02d",year % 100, month, day);
12. }
13. **else** **if**(day == 1)
14. {
15. printf("    %02d.%02d", month, day);
16. }
17. **else**
18. {
19. printf("%9d", day);
20. }
22. **if**(getDaySeqOfWeek(year, daySeqOfYear) == 6 || getDaySeqOfWeek(year, daySeqOfYear) == 0)
23. {
24. printf("!");
25. }
26. **else**
27. {
28. printf(" ");
29. }
30. **return**;
31. }
33. **void** printOneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow)
34. {
35. // Before Monday, print the week sequence
36. printf("[%02d]", weekSeqShow);
38. // Declare variables to navigate the semester
39. **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - getDaySeqOnJan1(year) + 2;
40. **int** sEndSeqOfYear = sStartSeqOfYear + 6;
41. **int** currentYearLength = isLeapYear(year);
42. **int** daySeqOfYear = sStartSeqOfYear;
44. // If this is the first week of semester, show full month and day for first day
45. **if** ( weekSeqShow == 1 )
46. {
47. printOneDay( year, daySeqOfYear, DATE\_INFO\_FULL );
48. daySeqOfYear ++;
49. }
51. // Print the semester calendar
52. **for** ( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )
53. {
54. // Print the month and day for the current day
55. **if** ( daySeqOfYear <= currentYearLength )
56. {
57. printOneDay( year, daySeqOfYear, DATE\_INFO\_BRIEF );
58. }
59. **else** **if**(daySeqOfYear == currentYearLength + 1)
60. {
61. printOneDay(year + 1, daySeqOfYear - currentYearLength, DATE\_INFO\_FULL);
62. }
63. **else**
64. {
65. printOneDay(year + 1, daySeqOfYear - currentYearLength, DATE\_INFO\_BRIEF);
66. }
67. }
69. // After Sunday, print a new line
70. printf("\n");
71. }

## 实验4代码：

main函数

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "date.h"
4. #include "fun.h"
5. #include "birthdayfun.h"

8. **int** main()
9. {
10. // Declare variables
11. **int** inputMonth, inputDay, Years[YEAR\_NUM];
13. // Declare a three-dimension array to record date information.
14. **int** Days[YEAR\_NUM][366][4] = {0};
16. // Display the program information
17. printf("Finding Birthday in year (%d-%d), please input the month and day : ", YEAR\_MIN, YEAR\_MAX);
18. scanf("%d%d", &inputMonth, &inputDay);
20. **if** (inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > 31)
21. {
22. printf("Sorry, the input month and day are invalid.\n");
23. **return** 1;
24. }
26. setYearArray(Years, YEAR\_NUM, YEAR\_MIN);
27. initialDays(Years, Days, YEAR\_NUM);
29. **int** totalNum, weekendNum, i;
30. weekendNum = 0;
31. totalNum = 0;
33. **for**(i = 0; i < YEAR\_NUM; i++)
34. {
35. printf("\n%s%d\n",  "Birthday in Year ", Years[i] );
37. **if**(isLeapYear(Years[i]) != 366 && inputDay == 29 && inputMonth == 2)
38. {
39. printf("Not found\n");
40. **continue**;
41. }
43. printf(" #M:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.");
44. **int** daySeqOfYear = getDaySeq(Years[i], inputMonth, inputDay);
45. **if**(Days[i][daySeqOfYear - 1][3] == 0 || Days[i - 1][daySeqOfYear][3] == 6)
46. {
47. weekendNum ++;
48. }
49. totalNum ++;
51. printOneWeek(Years[i], Days[i][daySeqOfYear - 1][2], i + 1);
52. }
54. printf("\nTotal %d birthday are found, %d of them are in weekends.\n", totalNum, weekendNum);

57. **return** 0;
58. }

date.h

1. #ifndef DATE\_H\_INCLUDED
2. #define DATE\_H\_INCLUDED


6. #endif // DATE\_H\_INCLUDED
7. **int** isLeapYear(**int** year);
8. **int** getDaySeqOnJan1(**int** year);
10. // Functions about month and day
11. **int** getMonthLength(**int** year, **int** month);
12. **int** getDaySeq(**int** year, **int** month, **int** day);
13. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day);
15. // Functions for properties of one day
16. **int** getMonth(**int** year, **int** daySeqOfYear);
17. **int** getDay(**int** year, **int** daySeqOfYear);
18. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear);
20. // Functions for day movement calculation
21. **int** getNextMonday(**int** year, **int** daySeqOfYear);
22. **int** getThisMonday(**int** year, **int** daySeqOfYear);
23. **int** getThisSunday(**int** year, **int** daySeqOfYear);

fun.h

1. #ifndef FUN\_H\_INCLUDED
2. #define FUN\_H\_INCLUDED


6. #endif // FUN\_H\_INCLUDED
8. #define YEAR\_NUM           9
9. #define YEAR\_MIN           2020
10. #define YEAR\_MAX           2028
11. #define DATE\_INFO\_BRIEF    1
12. #define DATE\_INFO\_FULL     2
14. **void** printOneDay(**int** year, **int** daySeqOfYear,**int** formatType);
15. **void** printOneWeek(**int** year, **int** weekSeqOfYear,**int** weekSeqShow);

birthday.h

1. #ifndef BIRTHDAYFUN\_H\_INCLUDED
2. #define BIRTHDAYFUN\_H\_INCLUDED


6. #endif // BIRTHDAYFUN\_H\_INCLUDED
7. **void** initialDays(**int** Years[], **int** Days[][366][4], **int** yearNum);
8. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin);

date.c

1. #include "date.h"
2. #include "fun.h"
3. #include "birthdayfun.h"
4. **const** **int** Month\_NORMAL\_YEAR[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
5. **const** **int** Month\_LEAP\_YEAR[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
6. **int** isLeapYear(**int** year)
7. {
8. **if** (year % 4 == 0 && year % 100 != 0)
9. {
10. **return** 366;
11. }
12. **else**
13. {
14. **return** 365;
15. }
16. }
17. **int** getDaySeqOnJan1(**int** year)
18. {
19. **int** day = 1;
20. **while** (year > 2024)
21. {
22. day += isLeapYear(year - 1);
23. year --;
24. }
25. **while**(year < 2024)
26. {
27. day -= isLeapYear(year);
28. year ++;
29. }
31. **if** (day > 0)
32. {
33. day %= 7;
34. }
35. **else**
36. {
37. day = day % 7 + 7;
38. }
39. **return** day;
40. }
42. **int** getMonthLength(**int** year, **int** month)
43. {
44. **int** day;
45. **switch**(month)
46. {
47. **case** 1:
48. day = 31;
49. **break**;
50. **case** 2:
51. **if** (year % 4 == 0 && year % 100 != 0)
52. {
53. day = 29;
54. }
55. **else**
56. {
57. day = 28;
58. }
59. **break**;
60. **case** 3:
61. day = 31;
62. **break**;
63. **case** 4:
64. day = 30;
65. **break**;
66. **case** 5:
67. day = 31;
68. **break**;
69. **case** 6:
70. day = 30;
71. **break**;
72. **case** 7:
73. day = 31;
74. **break**;
75. **case** 8:
76. day = 31;
77. **break**;
78. **case** 9 :
79. day = 30;
80. **break**;
81. **case** 10:
82. day = 31;
83. **break**;
84. **case** 11:
85. day = 30;
86. **break**;
87. **case** 12:
88. day = 31;
89. **break**;
90. }
92. **return** day;
93. }
94. **int** getDaySeq(**int** year, **int** month, **int** day)
95. {
96. /\*for(;month > 1 ;month --)
97. {
98. day += getMonthLength(year, month);
99. }\*/
100. **if**(isLeapYear(year) == 365)
101. {
102. **for**(;month > 1; month --)
103. {
104. day += Month\_NORMAL\_YEAR[month - 2];
105. }
106. }
107. **else**
108. {
109. **for**(;month > 1; month --)
110. {
111. day += Month\_LEAP\_YEAR[month - 2];
112. }
113. }
114. **return** day;
115. }
116. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day)
117. {
118. **int** daySeqOfYear = getDaySeq(year, month, day);
119. daySeqOfYear = daySeqOfYear + getDaySeqOnJan1(year) - 1;
121. **int** week;
122. **if**(daySeqOfYear % 7 == 0)
123. {
124. week = daySeqOfYear / 7;
125. }
127. **else**
128. {
129. week = daySeqOfYear / 7 + 1;
130. }
132. **return** week;
133. }
135. **int** getMonth(**int** year, **int** daySeqOfYear)
136. {
137. **int** month ;
138. **for**(month = 1; daySeqOfYear > getMonthLength(year, month); month ++)
139. {
140. daySeqOfYear -= getMonthLength(year, month);
141. }
142. **return** month;
143. }
144. **int** getDay(**int** year, **int** daySeqOfYear)
145. {
146. **int** month ;
147. **for**(month = 1;daySeqOfYear > getMonthLength(year, month); month ++)
148. {
149. daySeqOfYear -= getMonthLength(year, month);
150. }
151. **return** daySeqOfYear;
152. }
153. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
154. {
155. **int** day = (daySeqOfYear + getDaySeqOnJan1(year) - 1) % 7;
156. **return** day;
157. }
159. **int** getNextMonday(**int** year, **int** daySeqOfYear)
160. {
161. **int** day = (getDaySeqOfWeek(year, daySeqOfYear) == 1) ?
162. daySeqOfYear : daySeqOfYear + 8 - getDaySeqOfWeek(year, daySeqOfYear);
163. **return** day;
164. }
165. **int** getThisMonday(**int** year, **int** daySeqOfYear)
166. {
167. **int** day = daySeqOfYear - getDaySeqOfWeek(year, daySeqOfYear) + 1;
168. **return** day;
169. }
170. **int** getThisSunday(**int** year, **int** daySeqOfYear)
171. {
172. **int** day = (getDaySeqOfWeek(year,daySeqOfYear) == 0) ?
173. daySeqOfYear : daySeqOfYear + 7 - getDaySeqOfWeek(year,daySeqOfYear);
175. **return** day;
176. }

fun.c

1. #include "date.h"
2. #include "fun.h"
3. #include "birthdayfun.h"
5. **void** printOneDay(**int** year, **int** daySeqOfYear,**int** formatType)
6. {
7. **int** month = getMonth(year, daySeqOfYear);
8. **int** day = getDay(year, daySeqOfYear);
10. **if**(formatType == 2 )
11. {
12. printf(" %02d.%02d.%02d",year % 100, month, day);
13. }
14. **else** **if**(day == 1)
15. {
16. printf("    %02d.%02d", month, day);
17. }
18. **else**
19. {
20. printf("%9d", day);
21. }
23. **if**(getDaySeqOfWeek(year, daySeqOfYear) == 6 || getDaySeqOfWeek(year, daySeqOfYear) == 0)
24. {
25. printf("!");
26. }
27. **else**
28. {
29. printf(" ");
30. }
31. **return**;
32. }
34. **void** printOneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow)
35. {
36. // Before Monday, print the week sequence
37. printf("[%02d]", weekSeqShow);
39. // Declare variables to navigate the semester
40. **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - getDaySeqOnJan1(year) + 2;
41. **int** sEndSeqOfYear = sStartSeqOfYear + 6;
42. **int** currentYearLength = isLeapYear(year);
43. **int** daySeqOfYear = sStartSeqOfYear;
45. // If this is the first week of semester, show full month and day for first day
46. /\*if ( weekSeqShow == 1 )
47. {
48. printOneDay( year, daySeqOfYear, DATE\_INFO\_FULL );
49. daySeqOfYear ++;
50. }\*/
52. // Print the semester calendar
53. **for** ( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )
54. {
55. // Print the month and day for the current day
56. **if** ( daySeqOfYear <= currentYearLength )
57. {
58. printOneDay( year, daySeqOfYear, DATE\_INFO\_BRIEF );
59. }
60. **else** **if**(daySeqOfYear == currentYearLength + 1)
61. {
62. printOneDay(year + 1, daySeqOfYear - currentYearLength, DATE\_INFO\_FULL);
63. }
64. **else**
65. {
66. printOneDay(year + 1, daySeqOfYear - currentYearLength, DATE\_INFO\_BRIEF);
67. }
68. }
69. // After Sunday, print a new line
70. printf("\n");
71. }

birthday.c

1. #include "date.h"
2. #include "fun.h"
3. #include "birthdayfun.h"
5. **void** initialDays(**int** Years[], **int** Days[][366][4], **int** yearNum)
6. {
7. **int** year, month, day, yearLength, weekSeq, seqofweek;
8. **int** i, j;
10. **for** (i = 0; i < yearNum; i++)
11. {
12. year = Years[i];
13. yearLength = isLeapYear(year);
14. **for** (j = 0; j < yearLength; j++)
15. {
16. Days[i][j][0] = getMonth(year, j + 1);
17. Days[i][j][1] = getDay(year, j + 1);
18. Days[i][j][2] = getWeekSeqOfYear(year, Days[i][j][0], Days[i][j][1]);
19. Days[i][j][3] = getDaySeqOfWeek(year, j + 1);
20. }
21. }
23. **return**;
24. }
26. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin)
27. {
28. **int** i = 0;
29. **for**(i = 0; i < yearNum; i++)
30. {
31. Years[i] = yearMin + i;
32. }
34. **return** 0;
35. }

## 实验5代码：

1. #include<stdio.h>
2. #include<math.h>
4. #define YEAR\_NUM           9
5. #define YEAR\_MIN           2020
6. #define YEAR\_MAX           2028
7. #define DATE\_INFO\_BRIEF    1
8. #define DATE\_INFO\_FULL     2
9. #define DATE\_NOSHOW        3
10. #define DATE\_STAR          4
12. **typedef** **struct** main
13. {
14. **int** year;
15. **int** month;
16. **int** day;
17. **int** daySeq;
18. **int** weekSeq;
19. **int** weekDay;/\* data \*/
20. } Day;
21. **const** **int** Month\_NORMAL\_YEAR[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
22. **const** **int** Month\_LEAP\_YEAR[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
24. **int** isLeapYear(**int** year);
25. **int** getDaySeqOnJan1(**int** year);
27. // Functions about month and day
28. **int** getMonthLength(**int** year, **int** month);
29. **int** getDaySeq(**int** year, **int** month, **int** day);
30. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day);
32. // Functions for properties of one day
33. **int** getMonth(**int** year, **int** daySeqOfYear);
34. **int** getDay(**int** year, **int** daySeqOfYear);
35. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear);
37. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin);
39. **int** isDay(**int** year, **int** month, **int** day);
40. Day setDay(**int** year, **int** month, **int** day);
41. Day getDayBefore(Day currentday, **int** interval);
42. Day getDayAfter(Day currentday, **int** interval);
43. **int** getTwoDaysInterval(Day startDay, Day endDay);
44. **void** printDay(Day currentday, **int** displayFormat);



49. **int** main(**void**)
50. {
51. **int** i, inputMonth, inputDay, displayFormat;
52. Day birthday, prepareday,printStartDay,currentDay;
53. **int** printDayRange = 3;
54. **int** Years[YEAR\_NUM];
56. setYearArray(Years, YEAR\_NUM, YEAR\_MIN);
58. printf("Finding Birthday in year (%d-%d), please input the month and day : ", YEAR\_MIN, YEAR\_MAX);
59. scanf("%d%d", &inputMonth, &inputDay);
61. **if** (inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > 31)
62. {
63. printf("Sorry, the input month and day are invalid.\n");
64. **return** 1;
65. }

68. **for** ( i = 0 ; i < YEAR\_NUM ; i ++ )
69. {
70. printf( "\n%s%s%d\n", " ", "Birthday in Year ", Years[i] );
71. **if** ( isDay(Years[i], inputMonth, inputDay) == 1 )
72. {
73. printf(" Not found.\n");
74. **continue**;
75. }
76. birthday = setDay ( Years[i], inputMonth, inputDay );
77. prepareday = getDayBefore( birthday, printDayRange );
78. printStartDay = getDayBefore( prepareday, prepareday.weekDay );
79. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n[%02d]","Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.",printStartDay.weekSeq);
80. **for** ( currentDay = printStartDay; getTwoDaysInterval(currentDay, birthday) <= 0;
81. currentDay = getDayAfter( currentDay, 1 ) )
82. {
83. **if** ( getTwoDaysInterval(currentDay, prepareday) < 0 )
84. {
85. displayFormat = DATE\_NOSHOW;
86. }
87. **else** **if** ( getTwoDaysInterval(currentDay, prepareday) == 0 )
88. {
89. displayFormat = DATE\_INFO\_FULL;
90. }
91. **else** **if** ( getTwoDaysInterval(currentDay, birthday) < 0 )
92. {
93. displayFormat = DATE\_INFO\_BRIEF;
94. }
95. **else** **if** ( getTwoDaysInterval(currentDay, birthday) == 0 )
96. {
97. displayFormat = DATE\_STAR;
98. }
99. printDay(currentDay, displayFormat);
100. **if**(currentDay.weekDay == 6 && displayFormat != DATE\_STAR)
101. {
102. printf("\n #W:%10s%10s%10s%10s%10s%10s%10s\n[%02d]","Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.", birthday.weekSeq );
103. }
104. }
105. printf("\n");
106. }
107. }

110. **int** isLeapYear(**int** year)
111. {
112. **if** (year % 4 == 0 && year % 100 != 0)
113. {
114. **return** 366;
115. }
116. **else**
117. {
118. **return** 365;
119. }
120. }
121. **int** getDaySeqOnJan1(**int** year)
122. {
123. **int** day = 1;
124. **while** (year > 2024)
125. {
126. day += isLeapYear(year - 1);
127. year --;
128. }
129. **while**(year < 2024)
130. {
131. day -= isLeapYear(year);
132. year ++;
133. }
135. **if** (day > 0)
136. {
137. day = (day - 1) % 7;
138. }
139. **else**
140. {
141. day = (day - 1) % 7 + 7;
142. }
143. **return** day;
144. }
146. **int** getMonthLength(**int** year, **int** month)
147. {
148. **int** day;
149. **switch**(month)
150. {
151. **case** 1:
152. day = 31;
153. **break**;
154. **case** 2:
155. **if** (year % 4 == 0 && year % 100 != 0)
156. {
157. day = 29;
158. }
159. **else**
160. {
161. day = 28;
162. }
163. **break**;
164. **case** 3:
165. day = 31;
166. **break**;
167. **case** 4:
168. day = 30;
169. **break**;
170. **case** 5:
171. day = 31;
172. **break**;
173. **case** 6:
174. day = 30;
175. **break**;
176. **case** 7:
177. day = 31;
178. **break**;
179. **case** 8:
180. day = 31;
181. **break**;
182. **case** 9 :
183. day = 30;
184. **break**;
185. **case** 10:
186. day = 31;
187. **break**;
188. **case** 11:
189. day = 30;
190. **break**;
191. **case** 12:
192. day = 31;
193. **break**;
194. }
196. **return** day;
197. }
198. **int** getDaySeq(**int** year, **int** month, **int** day)
199. {
200. /\*for(;month > 1 ;month --)
201. {
202. day += getMonthLength(year, month);
203. }\*/
204. **if**(isLeapYear(year) == 365)
205. {
206. **for**(;month > 1; month --)
207. {
208. day += Month\_NORMAL\_YEAR[month - 2];
209. }
210. }
211. **else**
212. {
213. **for**(;month > 1; month --)
214. {
215. day += Month\_LEAP\_YEAR[month - 2];
216. }
217. }
218. **return** day;
219. }
220. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day)
221. {
222. **int** daySeqOfYear = getDaySeq(year, month, day);
223. daySeqOfYear = daySeqOfYear + getDaySeqOnJan1(year) - 1;
225. **int** week;
227. week = daySeqOfYear / 7 + 1;

230. **return** week;
231. }
233. **int** getMonth(**int** year, **int** daySeqOfYear)
234. {
235. **int** month ;
236. **for**(month = 1; daySeqOfYear > getMonthLength(year, month); month ++)
237. {
238. daySeqOfYear -= getMonthLength(year, month);
239. }
240. **return** month;
241. }
242. **int** getDay(**int** year, **int** daySeqOfYear)
243. {
244. **int** month ;
245. **for**(month = 1;daySeqOfYear > getMonthLength(year, month); month ++)
246. {
247. daySeqOfYear -= getMonthLength(year, month);
248. }
249. **return** daySeqOfYear;
250. }
251. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
252. {
253. **int** day = (daySeqOfYear + getDaySeqOnJan1(year) - 2) % 7;
254. **return** day;
255. }
257. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin)
258. {
259. **int** i = 0;
260. **for**(i = 0; i < yearNum; i++)
261. {
262. Years[i] = yearMin + i;
263. }
265. **return** ;
266. }
268. **int** isDay(**int** year, **int** month, **int** day)
269. {
270. **if**(isLeapYear(year) != 366 && month == 2 && day == 29)
271. {
272. **return** 1;
273. }
275. **return** 0;
276. }
277. Day setDay(**int** year, **int** month, **int** day)
278. {
279. Day d;
280. d.year = year;
281. d.month = month;
282. d.day = day;
283. d.daySeq = getDaySeq(year, month, day);
284. d.weekSeq = getWeekSeqOfYear(year, month, day);
285. d.weekDay = getDaySeqOfWeek(year, d.daySeq);
287. **return** d;
288. }
289. Day getDayBefore(Day currentday, **int** interval)
290. {
291. /\*if (interval > 28)
292. {
293. printf("Input error in function getDayBefore!");
294. return currentday;
295. }
297. currentday.day -= interval;
298. currentday.weekDay -= interval;
299. currentday.daySeq -= interval;
301. if(currentday.weekDay <= 6)
302. {
303. currentday.weekSeq -= (currentday.weekDay / 7 + 1);
304. currentday.weekDay = (currentday.weekDay % 7 + 7 ) % 7;
305. }
307. if(currentday.day < 1)
308. {
309. if(currentday.month == 1)
310. {
311. currentday.year --;
312. currentday.month = 12;
313. currentday.daySeq += isLeapYear(currentday.year);
314. currentday.day += 31;
315. currentday.weekSeq = getWeekSeqOfYear(currentday.year, currentday.month, currentday.day);
316. }
318. else
319. {
320. currentday.month --;
321. currentday.day += getMonthLength(currentday.year, currentday.month);
322. }
323. }\*/
324. currentday.daySeq -= interval;
325. **while**(currentday.daySeq <= 0)
326. {
327. currentday.year --;
328. currentday.daySeq += isLeapYear(currentday.year );
329. }
331. currentday.month = getMonth(currentday.year, currentday.daySeq);
332. currentday.day = getDay(currentday.year,currentday.daySeq);
333. currentday.weekDay = getDaySeqOfWeek(currentday.year, currentday.daySeq);
334. currentday.weekSeq = getWeekSeqOfYear(currentday.year, currentday.month, currentday.day);
336. **return** currentday;
337. }
338. Day getDayAfter(Day currentday, **int** interval)
339. {
340. /\*
342. //compute day, month and year in different cases
343. if (currentday.day > getMonthLength(currentday.year, currentday.month))
344. {
345. if(currentday.month == 12) // If the day is in the next year
346. {
347. currentday.month = 1;
348. currentday.day -= isLeapYear(currentday.year);
349. currentday.daySeq = currentday.day;
350. currentday.weekSeq = 1;
351. currentday.year ++;
352. }
353. else
354. {
355. currentday.month ++;
356. currentday.day -= getMonthLength(currentday.year, currentday.month);
357. }\*/
359. currentday.daySeq += interval;
360. **while**(currentday.daySeq > isLeapYear(currentday.year))
361. {
362. currentday.year ++;
363. currentday.daySeq -= isLeapYear(currentday.year - 1);
364. }
365. currentday.month = getMonth(currentday.year, currentday.daySeq);
366. currentday.day = getDay(currentday.year,currentday.daySeq);
367. currentday.weekDay = getDaySeqOfWeek(currentday.year, currentday.daySeq);
368. currentday.weekSeq = getWeekSeqOfYear(currentday.year, currentday.month, currentday.day);
370. **return** currentday;
371. }
372. **int** getTwoDaysInterval(Day startDay, Day endDay)
373. {
374. **int** interval = 0;
376. **while**(endDay.year != startDay.year)
377. {
378. **if**(endDay.year > startDay.year)
379. {
380. interval -= isLeapYear(startDay.year);
381. startDay.year ++;
382. }
383. **else**
384. {
385. interval += isLeapYear(startDay.year - 1);
386. startDay.year --;
387. }
388. }
390. interval += (startDay.daySeq - endDay.daySeq);
391. **return** interval;
392. }
393. **void** printDay(Day currentday, **int** displayFormat)
394. {
395. **switch** (displayFormat)
396. {
397. **case** 1:
398. printf("%10d", currentday.day);
399. **break**;
400. **case** 2:
401. printf("%4d.%02d.%02d", (currentday.year % 100), currentday.month, currentday.day);
402. **break**;
403. **case** 3:
404. printf("%10s", "");
405. **break**;
406. **case** 4:
407. printf("    %02d.%02d\*", currentday.month, currentday.day);
408. **break**;
409. }
411. **return**;
412. }