

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 | 李奇峰 | 学号 | U202411282 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2025.1.16 | 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

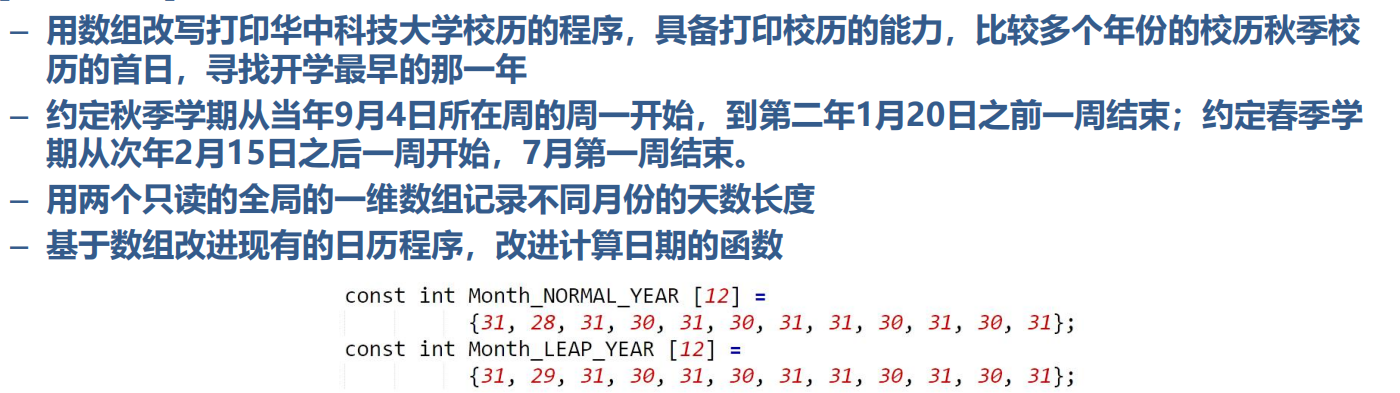
操作系统：Windows 10

编程工具：CodeBlocks 16.01

# 实验一

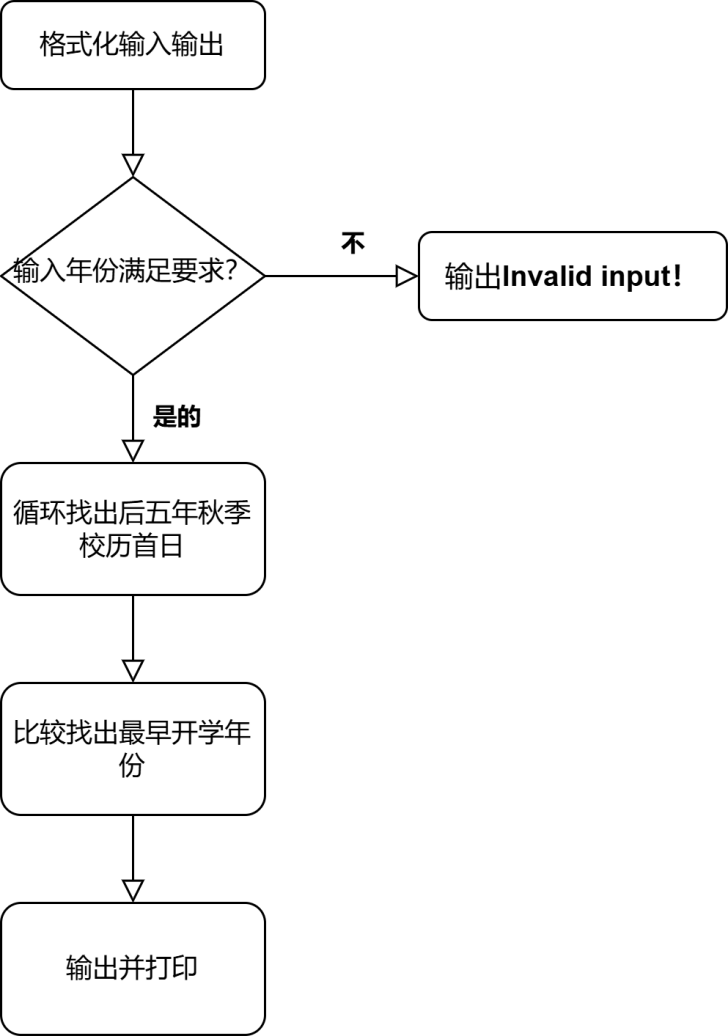
## 实验任务







## 实验步骤

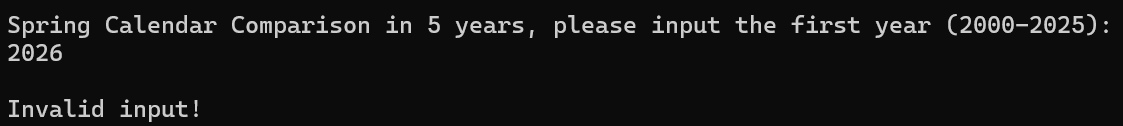
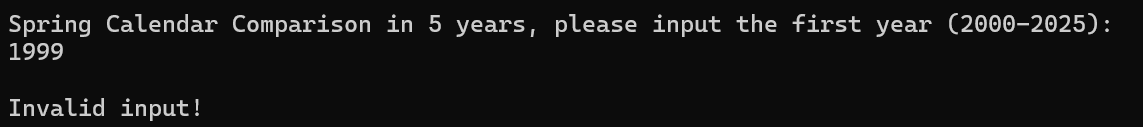


## 代码测试

### 测试点 1

测试思路：测试能否识别非法输入

预期结果：输出“Invalid input!”

实际结果：

### 测试点2

测试思路：测试能否正确找出秋季校历首日

预期结果：打印9月1日所在周历

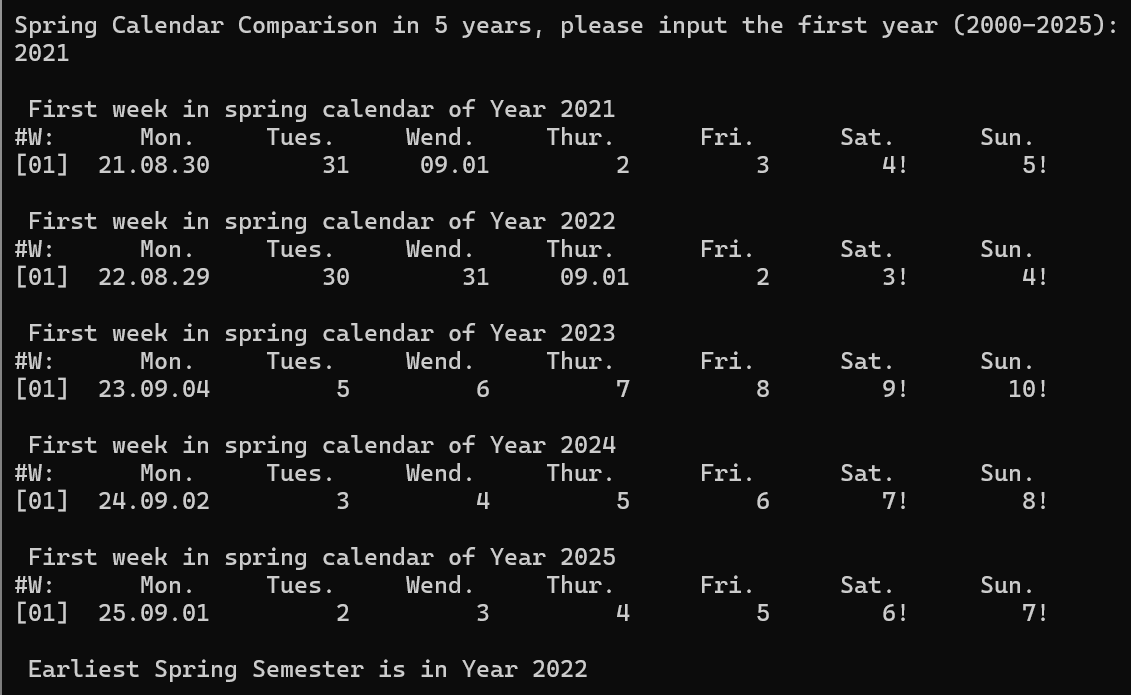
实际结果：

### 测试点3

测试思路：测试能否正确找出秋季校历最早开学日并打印

预期结果：输出最早开学日所在年份

实际结果：



## 实验结论

代码达到功能目标

## 实验总结

实验成功，但存在一定数组越界风险，后续需要注意。

# 实验二

## 实验任务





## 实验步骤

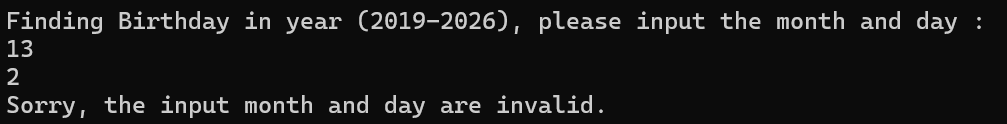


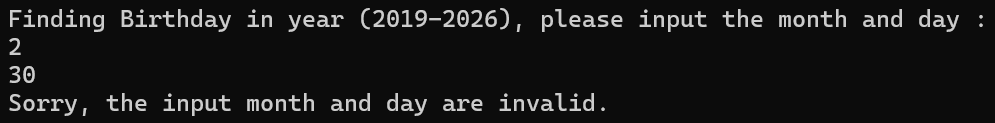
## 代码测试

### 测试点1

测试思路：测试能否判断并输出异常输入

预期结果：输出Sorry, the input month and day are invalid.

实际结果：

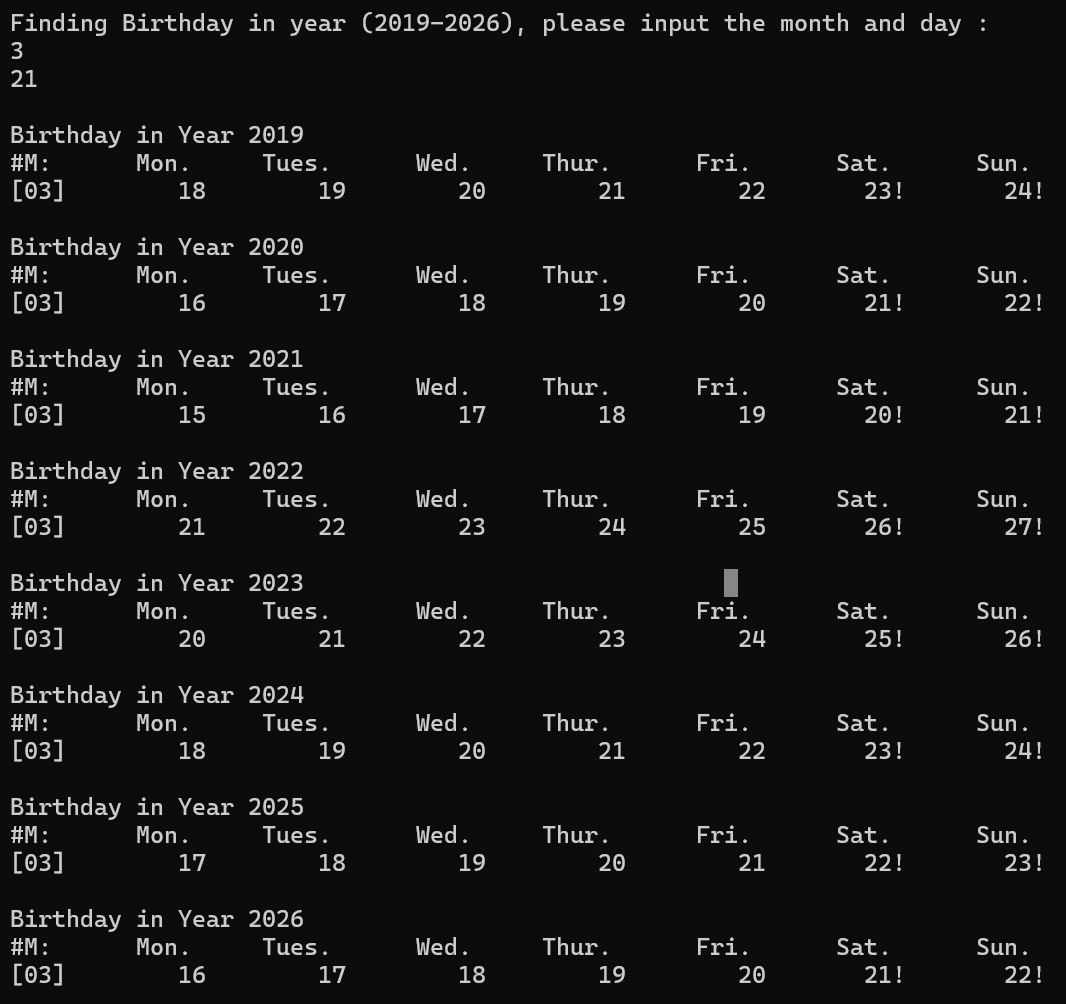


### 测试点2

测试思路：测试能否正常打印生日所在周周历

预期结果：依年每行打印周历

实际结果：

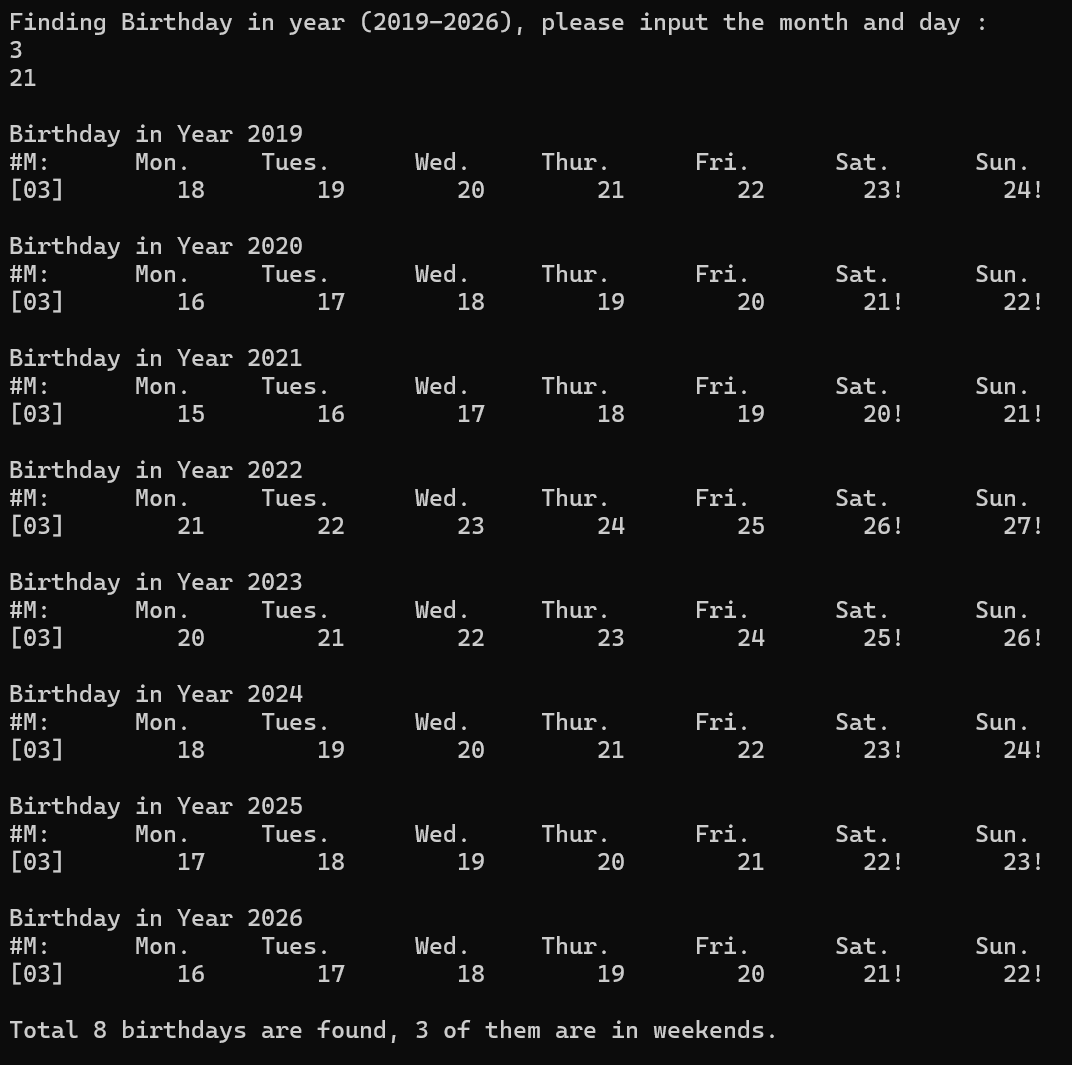


### 测试点3

测试思路：测试能否正常计算生日在周末的次数

预期结果：输出累加结果

实际结果：

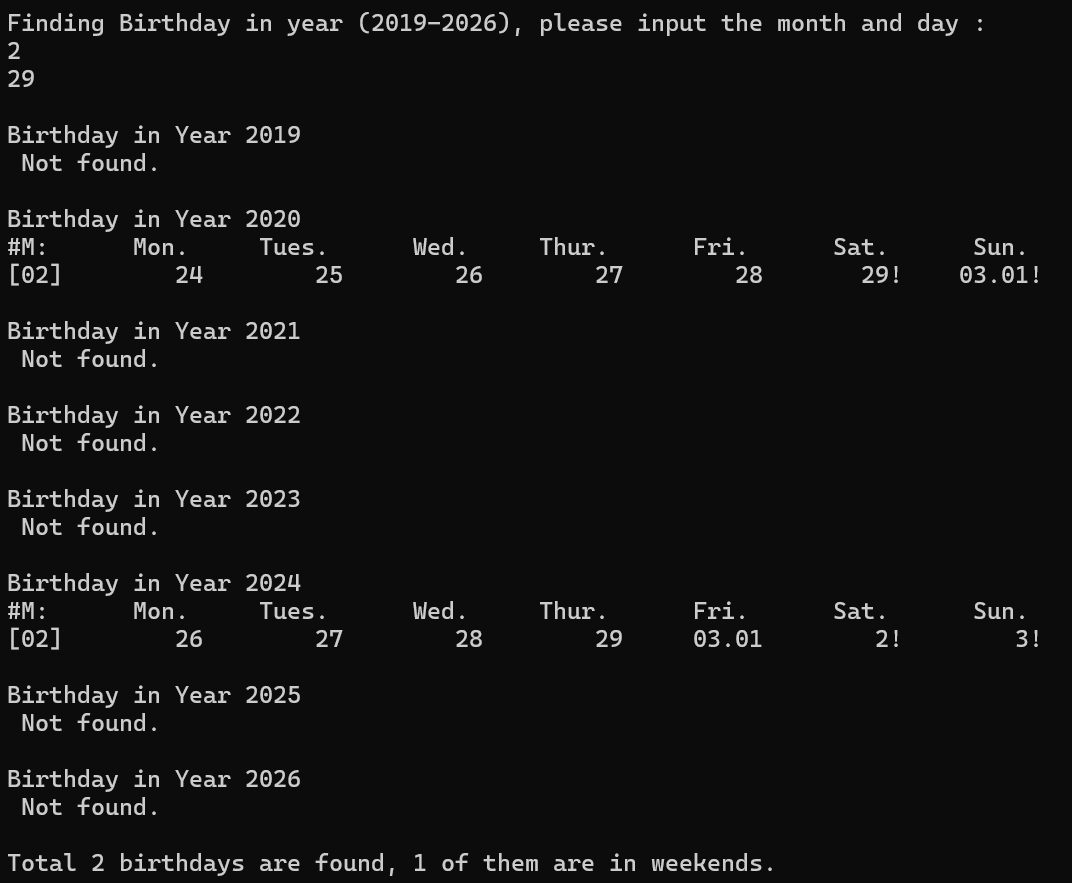


### 测试点4

测试思路：测试2月29日的特例

预期结果：闰年正常输出而平年输出Not found

实际结果：



## 实验结论

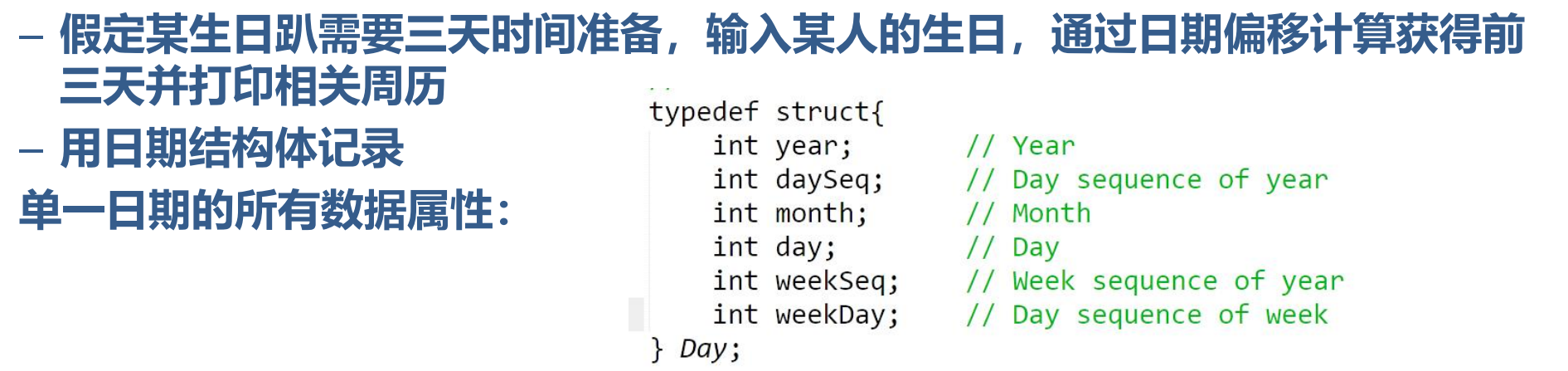
代码达到功能目标

## 实验总结

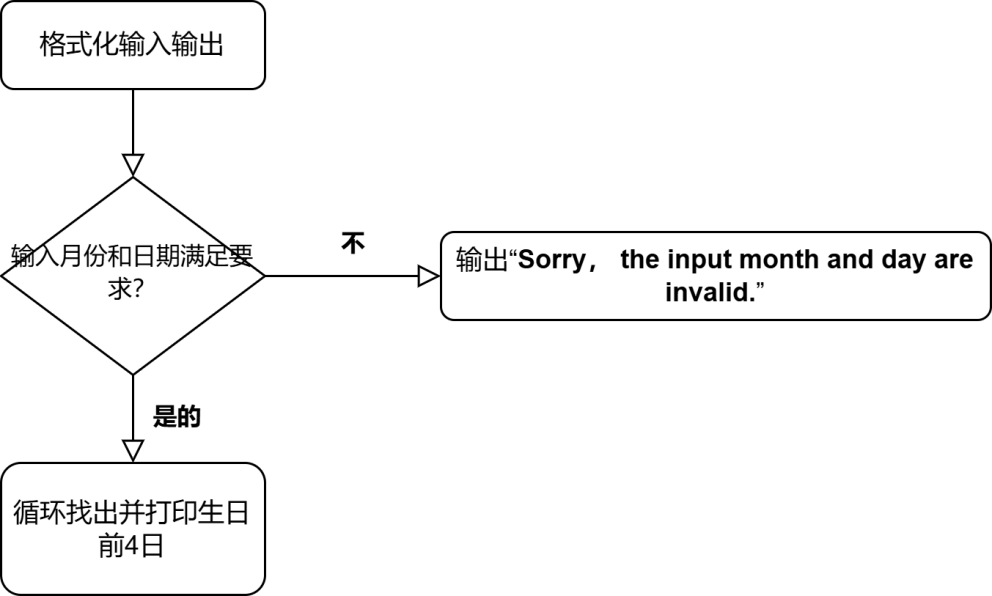
实验成功，但也提醒我们要注意特例。

# 实验三

## 试验任务



## 实验步骤



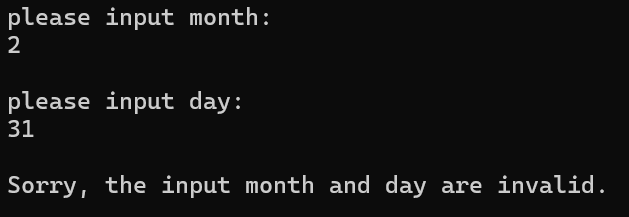
## 代码测试

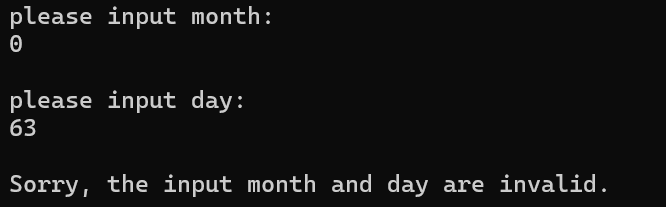
### 测试点1

测试思路：测试能否判断并输出异常输入

预期结果：输出Sorry, the input month and day are invalid.

实际结果：



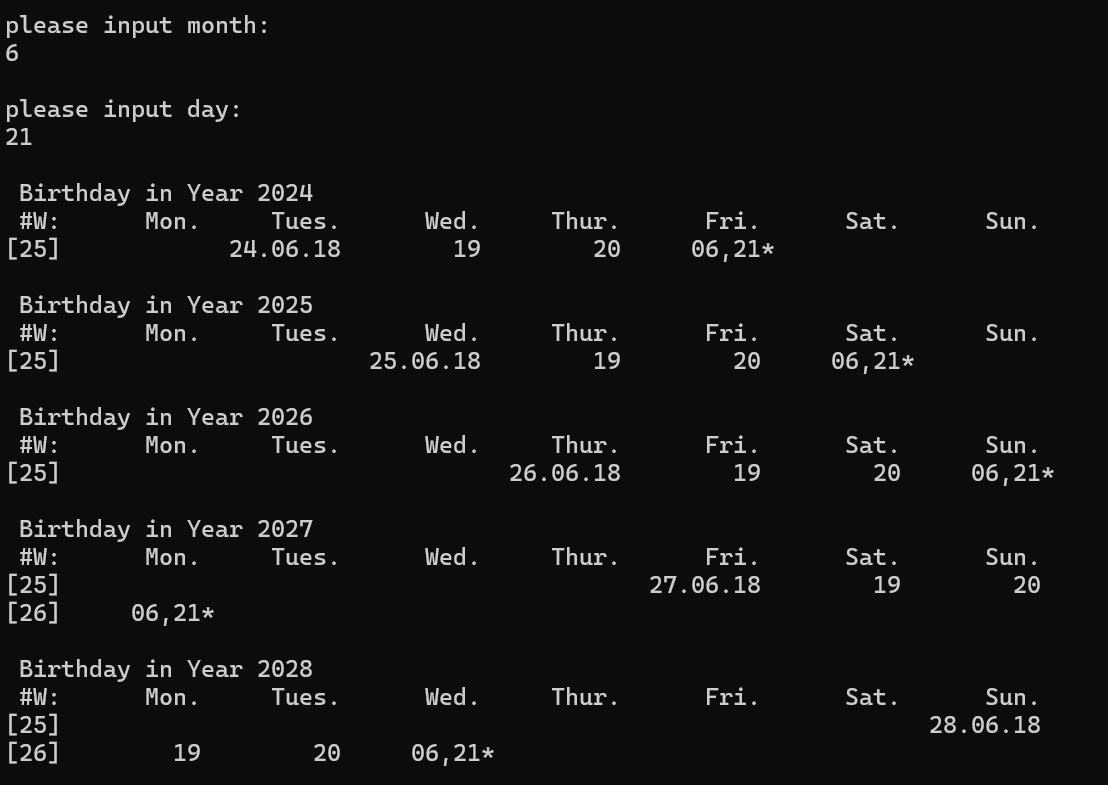


### 测试点2

测试思路：测试能否正常打印生日前4日日历

预期结果：依年每行打印日历

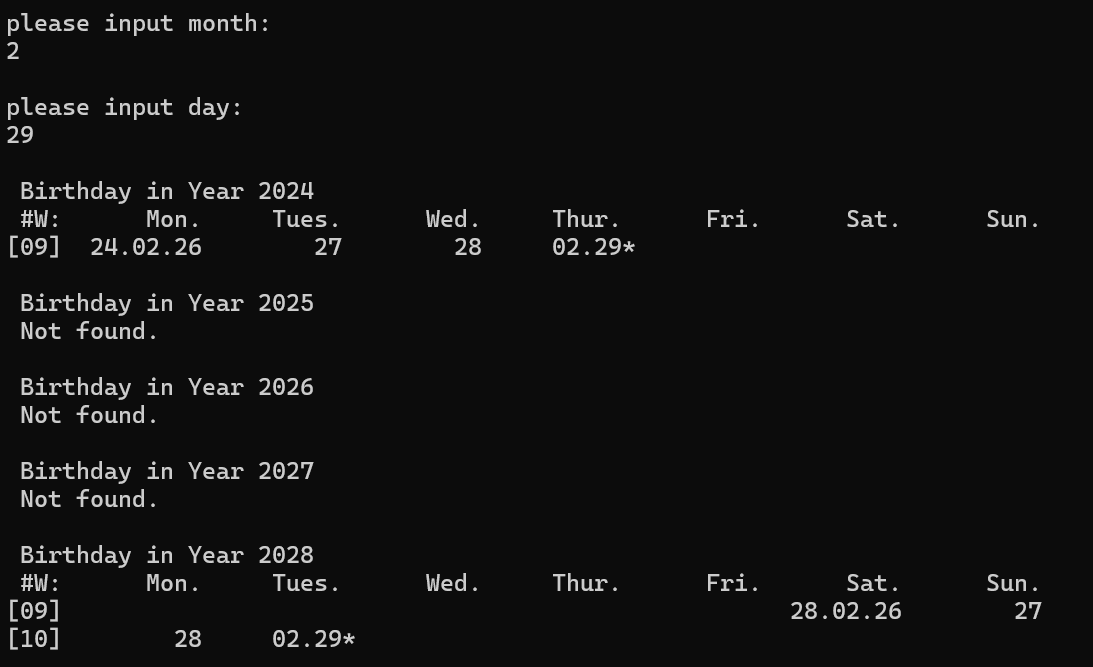
实际结果：



测试思路：测试2月29日的特例

预期结果：闰年正常输出而平年输出Not found

实际结果：



## 实验结论

代码达到功能目标

## 实验总结

实验成功，结构体使用极大减少了编程量。

# 本课程学习总结

## 学习C语言的历程和总结（心酸血泪史）

从日历系列出发，从控制语句到函数再到数组、结构体，整体成循序渐进、由易到难的趋势。过程中每一次任务都是对课堂内容的整体复习和对后续任务的铺垫。而其中为数函数部分耗时最久，具体表现为对每一个函数的细细斟酌、debug时的崩溃和后续任务中因要求改变而对函数进行全面优化。但是回首看来，正是这一遍遍的修改让我清晰了最优解的重要性，让我明白了思路和逻辑的重要性。

## 程序中出现的问题和改进

日历系列的函数部分常常出现逻辑不通的问题，要求我们今后要先对问题有一个整体上的解决思路再一步步落实到具体函数。

数组则要考虑是否存在越界问题，要对特殊情况逆向思考。

## 代码规范与调试技巧

代码编写时应该注意空格，尽量一行只存放一句话，必要时可以额外空一行出来以保持代码的美观度和可读性。同时要习惯于使用函数来简化代码，养成写注释的习惯。在编程过程中要一步一个脚印，对于写好的部分要求实时debug

# 附录

1. 实验一

* main.c

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "date.h"
4. #include "funs.h"
6. **const** **int** Month\_NORMAL\_YEAR [12] =
7. {31,28,31,30,31,30,31,31,30,31,30,31};
9. **const** **int** Month\_LEAP\_YEAR [12] =
10. {31,29,31,30,31,30,31,31,30,31,30,31};
12. **int** main()
13. {
14. **int** year ,Years[ YEAR\_NUM ];
16. printf("Spring Calendar Comparison in %d years, please input the first year (%d-%d): \n",
17. YEAR\_NUM, YEAR\_MIN, YEAR\_MAX - YEAR\_NUM ) ;
19. scanf("%d", &year) ;
21. **if**( year < YEAR\_MIN || year + YEAR\_NUM > YEAR\_MAX )
22. {
23. printf("\nInvalid input!\n") ;
24. **return** 0 ;
25. }
27. setYearArray( Years, YEAR\_NUM, year );
29. **int** i;
30. **int** startDays[ YEAR\_NUM ] = { 0 } ;
32. **for**( i = 0 ; i< YEAR\_NUM ; i ++ )
33. {
34. printf("\n%s%s%d\n"," ","First week in spring calendar of Year ", Years[i]) ;
35. printf("#W:%10s%10s%10s%10s%10s%10s%10s\n"
36. ,"Mon.","Tues.","Wend.","Thur.","Fri.","Sat.","Sun.") ;
37. printoneWeek( Years[ i ], getWeekSeqOfYear( Years[ i ], 9, 4 ), 1) ;
38. startDays[ i ] = getThisMonday( Years[ i ], getDaySeq( Years[ i ], 9, 4 )) ;
39. }
41. **int** min = 0 ;
43. **for**( i = 1 ; i < YEAR\_NUM ; i ++)
44. {
45. min = ( startDays[ i ] < startDays[ min ]) ? i : min ;
46. }
48. printf("\n%s%s%d\n"," " ,"Earliest Spring Semester is in Year " , Years[ min ] );
50. **return** 0 ;
51. }

* Funs.c

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. **void** printOneDay( **int** year, **int** daySeqOfYear , **int** formatType )
8. {
9. **int** day = getDay( year, daySeqOfYear);
10. **int** month = getMonth( year, daySeqOfYear);
11. **int** week = getDaySeqOfWeek( year, daySeqOfYear);
13. **if**( formatType == 1 )
14. {
15. printf("%2s%02d.%02d.%02d"," ", year % 100 , month, day);
16. }
18. **else** **if**( day == 1)
19. {
20. **if**( month == 1)
21. {
22. **if**( week == 6|| week == 0)
23. {
24. printf("%1s%02d.%02d.%02d!"," ", year % 100 , month, day);
25. }
27. **else**
28. {
29. printf("%2s%02d.%02d.%02d"," ", year % 100 , month, day);
30. }
31. }
33. **else** **if**( week == 6|| week == 0)
34. {
35. printf("%4s%02d.%02d!"," ", month, day);
36. }
38. **else**
39. {
40. printf("%5s%02d.%02d"," ", month, day);
41. }
42. }
44. **else**
45. {
46. **if**( week == 6|| week == 0)
47. {
48. printf("%9d!", day);
49. }
51. **else**
52. {
53. printf("%10d", day);
54. }
55. }
56. }
58. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow )
59. {
61. printf("[%02d]", weekSeqShow) ;
63. **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - getDaySeqOnJan1( year ) + 2 ;
64. **int** sEndSeqOfYear = sStartSeqOfYear + 6 ;
65. **int** currentyear = 365 + isLeapYear( year ) ;
66. **int** daySeqOfYear = sStartSeqOfYear ;
68. **if**( weekSeqShow == 1)
69. {
70. printOneDay( year, daySeqOfYear , DATE\_INFO\_FULL ) ;
72. daySeqOfYear ++ ;
73. }

76. **for**( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )
77. {
78. **if**( daySeqOfYear <= currentyear )
79. {
80. printOneDay( year, daySeqOfYear , DATE\_INFO\_BRIEF ) ;
81. }
83. **else**
84. {
85. printOneDay( year + 1, daySeqOfYear - currentyear , DATE\_INFO\_BRIEF ) ;
86. }
88. }
90. printf("\n") ;
91. }

* Dates.c

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. #define MONTH\_NUM 12
9. **int** getDay( **int** year, **int** daySeqOfYear )
10. {
11. **extern** **int**  Month\_LEAP\_YEAR[12];
12. **extern** **int**  Month\_NORMAL\_YEAR[12];
14. **int** i = 0 ;
15. **int** ret = daySeqOfYear ;
17. **if**( isLeapYear( year ) == 1 )
18. {
19. **while**( ret > Month\_LEAP\_YEAR[ i ] )
20. {
21. ret -= Month\_LEAP\_YEAR[ i ] ;
23. i ++ ;
24. }
25. }
27. **else**
28. {
29. **while**( ret > Month\_NORMAL\_YEAR[ i ] )
30. {
31. ret -= Month\_NORMAL\_YEAR[ i ] ;
33. i ++ ;
34. }
35. }
37. **return**  ret ;
38. }
40. **int** getDaySeq( **int** year, **int** month, **int** day )
41. {
42. **switch**( month )
43. {
44. **case** 12:
45. day += 30 ;
47. **case** 11:
48. day += 31 ;
50. **case** 10:
51. day += 30 ;
53. **case** 9:
54. day += 31 ;
56. **case** 8:
57. day += 31 ;
59. **case** 7:
60. day += 30 ;
62. **case** 6:
63. day += 31 ;
65. **case** 5:
66. day += 30 ;
68. **case** 4:
69. day += 31 ;
71. **case** 3:
72. day += 28 + isLeapYear( year ) ;
74. **case** 2:
75. day += 31 ;
77. **break** ;
78. }
80. **return** day ;
81. }
83. **int** getDaySeqOfWeek( **int** year, **int** daySeqOfYear )
84. {
86. daySeqOfYear += getDaySeqOnJan1( year ) - 1 ;
88. daySeqOfYear = daySeqOfYear % 7 ;
90. **return** daySeqOfYear ;
91. }
93. **int** getDaySeqOnJan1( **int** year )
94. {
95. **int** result ;
97. result = ( year - 1 +( year - 1) / 4 - ( year - 1)/ 100 +
98. ( year - 1)/ 400) % 7 + 1 ;
100. **return** result ;
101. }
103. **int** getMonth( **int** year, **int** dayseq )
104. {
105. **extern** **int**  Month\_LEAP\_YEAR[12];
106. **extern** **int**  Month\_NORMAL\_YEAR[12];
108. **int** temp = 1 ;
110. **if**( isLeapYear( year ) == 1 )
111. {
112. **while**( dayseq > Month\_LEAP\_YEAR[ temp - 1 ] )
113. {
114. dayseq -= Month\_LEAP\_YEAR[ temp - 1 ] ;
116. temp ++ ;
117. }
118. }
120. **else**
121. {
122. **while**( dayseq > Month\_NORMAL\_YEAR[ temp - 1 ] )
123. {
124. dayseq -= Month\_NORMAL\_YEAR[ temp - 1 ] ;
126. temp ++ ;
127. }
128. }
129. **return** temp ;
130. }
132. **int** getNextMonday( **int** year, **int** day )
133. {
134. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 1)
135. {
136. **if**(( day + getDaySeqOnJan1( year ) - 1 ) % 7 == 0)
137. {
138. day ++ ;
139. }
141. **else**
142. {
143. day += (7 -( day + getDaySeqOnJan1( year ) - 1 ) % 7) + 1 ;
144. }
145. }
147. **return** day ;
148. }
150. **int** getThisMonday ( **int** year, **int** day )
151. {
152. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 1)
153. {
154. **if**(( day + getDaySeqOnJan1( year ) - 1 ) % 7 == 0)
155. {
156. day -= 6 ;
157. }
159. **else**
160. {
161. day -= ( day + getDaySeqOnJan1( year ) - 1 ) % 7 - 1 ;
162. }
163. }
165. /\* day = getNextMonday ( year, day) - 7 ; \*/
167. **return** day ;
168. }
170. **int** getThisSunday( **int** year, **int** day )
171. {
172. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 0)
173. {
174. day += 7 -(day + getDaySeqOnJan1( year ) - 1 ) % 7 ;
175. }
177. **return** day ;
178. }
180. **int** isLeapYear( **int** year )
181. {
182. **if**( ( year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )
183. {
184. **return** 1 ;
185. }
187. **return** 0 ;
188. }
190. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day )
191. {
192. **int** week ;
194. **int** daySeqOfYear = getDaySeq( year, month, day ) ;
196. **int** currentyear = 365 + isLeapYear( year ) ;
198. **if**( daySeqOfYear > currentyear )
199. {
200. **if**( getDaySeqOfWeek( year + 1, daySeqOfYear - currentyear) == 0)
201. {
202. week = ( daySeqOfYear + getDaySeqOnJan1( year ) ) / 7 ;
203. }
205. **else**
206. {
207. week = ( daySeqOfYear + getDaySeqOnJan1( year ) +
208. ( 7 - getDaySeqOfWeek( year + 1, daySeqOfYear - currentyear))) / 7 ;
209. }
210. }
212. **else**
213. {
214. **if**( getDaySeqOfWeek( year , daySeqOfYear ) == 0)
215. {
216. week = ( daySeqOfYear + getDaySeqOnJan1( year ) ) / 7 ;
217. }
219. **else**
220. {
221. week = ( daySeqOfYear + getDaySeqOnJan1( year ) +
222. ( 7 - getDaySeqOfWeek( year , daySeqOfYear)))/ 7 ;
223. }
224. }
226. **return** week;
227. }
229. **void** setYearArray( **int** Years[], **int** yearnum, **int** year )
230. {
231. **int** num1 ;
233. **for**( num1 = 0; num1 < yearnum ; num1 ++ , year ++)
234. {
235. Years[ num1 ] = year;
236. }
237. }

* Funs.h

1. #ifndef FUNS\_H\_INCLUDED
2. #define FUNS\_H\_INCLUDED
4. **void** printOneDay( **int** year, **int** daySeqOfYear , **int** formatType );
5. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow );
7. #endif // FUNS\_H\_INCLUDED

* Dates.h

1. #ifndef DATE\_H\_INCLUDED
2. #define DATE\_H\_INCLUDED
4. #define YEAR\_MIN 2000
5. #define YEAR\_MAX 2030
6. #define YEAR\_NUM 5
7. #define DATE\_INFO\_BRIEF 0
8. #define DATE\_INFO\_FULL 1
10. **int** isLeapYear( **int** year );
11. **int** getDaySeqOnJan1( **int** year );
13. **int** getDaySeq( **int** year, **int** month, **int** day );
14. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day );
16. **int** getMonth( **int** year, **int** daySeqOfYear );
17. **int** getDay( **int** year, **int** daySeqOfYear );
18. **int** getDaySeqOfWeek( **int** year,**int** daySeqOfYear );
20. **int** getNextMonday( **int** year, **int** daySeqOfYear );
21. **int** getThisMonday ( **int** year, **int** day );
22. **int** getThisSunday( **int** year, **int** daySeqOfYear );
24. **void** setYearArray( **int** Years[], **int** yearnum, **int** year );
26. #endif // DATE\_H\_INCLUDED
27. 实验二

* main.c

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "date.h"
4. #include "funs.h"
6. **const** **int** Month\_NORMAL\_YEAR [12] =
7. {31,28,31,30,31,30,31,31,30,31,30,31};
9. **const** **int** Month\_LEAP\_YEAR [12] =
10. {31,29,31,30,31,30,31,31,30,31,30,31};
12. **int** main()
13. {
14. **int** inputMonth, inputDay, Years[YEAR\_NUM];
16. printf("Finding Birthday in year (%d-%d), please input the month and day : \n", YEAR\_MIN, YEAR\_MAX );
18. scanf("%d%d", &inputMonth, &inputDay );
20. **if** ( inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > Month\_LEAP\_YEAR [inputMonth - 1])
21. {
22. printf("Sorry, the input month and day are invalid.\n");
23. **return** 0;
24. }
26. setYearArray ( Years, YEAR\_NUM, YEAR\_MIN );
28. **int** Days[YEAR\_NUM][366][4] = {0};
30. initialDays ( Years, Days, YEAR\_NUM);
32. **int** totalNum = 0;
33. **int** weekendNum = 0;
35. **int** i,j;
36. **for** (i = 0; i < YEAR\_NUM; i ++)
37. {
38. **int** found = 0;
39. printf("\n%sBirthday in Year %d\n", "", Years[i]);
41. **for** (j = 0; j < (isLeapYear(Years[i]) ? 366 : 365); j ++)
42. {
43. **if** (Days[i][j][0] == inputMonth && Days[i][j][1] == inputDay)
44. {
45. totalNum ++;
46. found = 1;
47. printf("#M:%10s%10s%10s%10s%10s%10s%10s\n",
48. "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
50. printoneWeek(Years[i], getWeekSeqOfYear(Years[i], inputMonth, inputDay), inputMonth);
52. **if** (Days[i][j][3] == 6 || Days[i][j][3] == 0)
53. {
54. weekendNum++;
55. }
56. **break**;
57. }
58. }
60. **if** (!found)
61. {
62. printf(" Not found.\n");
63. }
64. }
65. printf("\nTotal %d birthdays are found, %d of them are in weekends.\n", totalNum, weekendNum);
66. **return** 0;
67. }

* Funs.c

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. **void** printOneDay( **int** year, **int** daySeqOfYear , **int** formatType )
8. {
9. **int** day = getDay( year, daySeqOfYear);
10. **int** month = getMonth( year, daySeqOfYear);
11. **int** week = getDaySeqOfWeek( year, daySeqOfYear);
13. **if**(day <= 0)
14. {
15. **if**( week == 6|| week == 0)
16. {
17. printf("%4s%02d.%02d!"," ", 12, day + 31);
18. }
20. **else**
21. {
22. printf("%5s%02d.%02d"," ", 12, day + 31);
23. }
24. }
25. **else** **if**( formatType == 1 )
26. {
27. printf("%2s%02d.%02d.%02d"," ", year % 100 , month, day);
28. }
29. **else** **if**( day == 1)
30. {
31. **if**( month == 1)
32. {
33. **if**( week == 6|| week == 0)
34. {
35. printf("%1s%02d.%02d.%02d!"," ", year % 100 , month, day);
36. }
38. **else**
39. {
40. printf("%2s%02d.%02d.%02d"," ", year % 100 , month, day);
41. }
42. }
44. **else** **if**( week == 6|| week == 0)
45. {
46. printf("%4s%02d.%02d!"," ", month, day);
47. }
49. **else**
50. {
51. printf("%5s%02d.%02d"," ", month, day);
52. }
53. }
55. **else**
56. {
57. **if**( week == 6|| week == 0)
58. {
59. printf("%9d!", day);
60. }
62. **else**
63. {
64. printf("%10d", day);
65. }
66. }
67. }
69. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow )
70. {
72. printf("[%02d]", weekSeqShow) ;
74. **int** sStartSeqOfYear = 7 \* ( weekSeqOfYear - 1 ) - getDaySeqOnJan1( year ) + 2 ;
75. **int** sEndSeqOfYear = sStartSeqOfYear + 6 ;
76. **int** currentyear = 365 + isLeapYear( year ) ;
77. **int** daySeqOfYear = sStartSeqOfYear ;
79. **if**( weekSeqShow == 1)
80. {
81. printOneDay( year, daySeqOfYear , DATE\_INFO\_FULL ) ;
83. daySeqOfYear ++ ;
84. }

87. **for**( ; daySeqOfYear <= sEndSeqOfYear ; daySeqOfYear ++ )
88. {
89. **if**( daySeqOfYear <= currentyear )
90. {
91. printOneDay( year, daySeqOfYear , DATE\_INFO\_BRIEF ) ;
92. }
94. **else**
95. {
96. printOneDay( year + 1, daySeqOfYear - currentyear , DATE\_INFO\_BRIEF ) ;
97. }
99. }
101. printf("\n") ;
102. }

* Dates.c

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. #define MONTH\_NUM 12
9. **int** getDay( **int** year, **int** daySeqOfYear )
10. {
11. **extern** **int**  Month\_LEAP\_YEAR[12];
12. **extern** **int**  Month\_NORMAL\_YEAR[12];
14. **int** i = 0 ;
15. **int** ret = daySeqOfYear ;
17. **if**( isLeapYear( year ) == 1 )
18. {
19. **while**( ret > Month\_LEAP\_YEAR[ i ] )
20. {
21. ret -= Month\_LEAP\_YEAR[ i ] ;
23. i ++ ;
24. }
25. }
27. **else**
28. {
29. **while**( ret > Month\_NORMAL\_YEAR[ i ] )
30. {
31. ret -= Month\_NORMAL\_YEAR[ i ] ;
33. i ++ ;
34. }
35. }
37. **return**  ret ;
38. }
40. **int** getDaySeq( **int** year, **int** month, **int** day )
41. {
42. **switch**( month )
43. {
44. **case** 12:
45. day += 30 ;
47. **case** 11:
48. day += 31 ;
50. **case** 10:
51. day += 30 ;
53. **case** 9:
54. day += 31 ;
56. **case** 8:
57. day += 31 ;
59. **case** 7:
60. day += 30 ;
62. **case** 6:
63. day += 31 ;
65. **case** 5:
66. day += 30 ;
68. **case** 4:
69. day += 31 ;
71. **case** 3:
72. day += 28 + isLeapYear( year ) ;
74. **case** 2:
75. day += 31 ;
77. **break** ;
78. }
80. **return** day ;
81. }
83. **int** getDaySeqOfWeek( **int** year, **int** daySeqOfYear )
84. {
86. daySeqOfYear += getDaySeqOnJan1( year ) - 1 ;
88. daySeqOfYear = daySeqOfYear % 7 ;
90. **return** daySeqOfYear ;
91. }
93. **int** getDaySeqOnJan1( **int** year )
94. {
95. **int** result ;
97. result = ( year - 1 +( year - 1) / 4 - ( year - 1)/ 100 +( year - 1)/ 400) % 7 + 1 ;
99. **return** result ;
100. }
102. **int** getMonth( **int** year, **int** dayseq )
103. {
104. **extern** **int**  Month\_LEAP\_YEAR[12];
105. **extern** **int**  Month\_NORMAL\_YEAR[12];
107. **int** temp = 1 ;
109. **if**( isLeapYear( year ) == 1 )
110. {
111. **while**( dayseq > Month\_LEAP\_YEAR[ temp - 1 ] )
112. {
113. dayseq -= Month\_LEAP\_YEAR[ temp - 1 ] ;
115. temp ++ ;
116. }
117. }
119. **else**
120. {
121. **while**( dayseq > Month\_NORMAL\_YEAR[ temp - 1 ] )
122. {
123. dayseq -= Month\_NORMAL\_YEAR[ temp - 1 ] ;
125. temp ++ ;
126. }
127. }
128. **return** temp ;
129. }
131. **int** getNextMonday( **int** year, **int** day )
132. {
133. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 1)
134. {
135. **if**(( day + getDaySeqOnJan1( year ) - 1 ) % 7 == 0)
136. {
137. day ++ ;
138. }
140. **else**
141. {
142. day += (7 -( day + getDaySeqOnJan1( year ) - 1 ) % 7) + 1 ;
143. }
144. }
146. **return** day ;
147. }
149. **int** getThisMonday ( **int** year, **int** day )
150. {
151. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 1)
152. {
153. **if**(( day + getDaySeqOnJan1( year ) - 1 ) % 7 == 0)
154. {
155. day -= 6 ;
156. }
158. **else**
159. {
160. day -= ( day + getDaySeqOnJan1( year ) - 1 ) % 7 - 1 ;
161. }
162. }
164. **return** day ;
165. }
167. **int** getThisSunday( **int** year, **int** day )
168. {
169. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 0)
170. {
171. day += 7 -(day + getDaySeqOnJan1( year ) - 1 ) % 7 ;
172. }
174. **return** day ;
175. }
177. **int** isLeapYear( **int** year )
178. {
179. **if**( ( year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )
180. {
181. **return** 1 ;
182. }
184. **return** 0 ;
185. }
187. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day )
188. {
189. **int** week ;
191. **int** daySeqOfYear = getDaySeq( year, month, day ) ;
193. **int** currentyear = 365 + isLeapYear( year ) ;
195. **if**( daySeqOfYear > currentyear )
196. {
197. **if**( getDaySeqOfWeek( year + 1, daySeqOfYear - currentyear) == 0)
198. {
199. week = ( daySeqOfYear + getDaySeqOnJan1( year ) ) / 7 ;
200. }
202. **else**
203. {
204. week = ( daySeqOfYear + getDaySeqOnJan1( year ) +
205. ( 7 - getDaySeqOfWeek( year + 1, daySeqOfYear - currentyear))) / 7 ;
206. }
207. }
209. **else**
210. {
211. **if**( getDaySeqOfWeek( year , daySeqOfYear ) == 0)
212. {
213. week = ( daySeqOfYear + getDaySeqOnJan1( year ) ) / 7 ;
214. }
216. **else**
217. {
218. week = ( daySeqOfYear + getDaySeqOnJan1( year ) +
219. ( 7 - getDaySeqOfWeek( year , daySeqOfYear)))/ 7 ;
220. }
221. }
223. **return** week;
224. }
226. **void** setYearArray( **int** Years[], **int** yearnum, **int** year )
227. {
228. **int** num1 ;
230. **for**( num1 = 0; num1 < yearnum ; num1 ++ , year ++)
231. {
232. Years[ num1 ] = year;
233. }
234. }
236. **void** initialDays(**int** Years[], **int** Days[][366][4], **int** yearNum )
237. {
239. **int** year, month, day, yearLength, weekSeq, seqOfWeek;
240. **int** i, j;
242. **for** (i = 0; i < YEAR\_NUM; i ++)
243. {
244. year = Years[i];
245. yearLength = isLeapYear( year ) ? 366 : 365;
247. **for**(j = 0; j < yearLength; j ++)
248. {
249. Days[i][j][0] = getMonth( year, j + 1);
250. Days[i][j][1] = getDay( year, j + 1 );
251. Days[i][j][2] = getWeekSeqOfYear( year, Days[i][j][0], Days[i][j][1] );
252. Days[i][j][3] = getDaySeqOfWeek( year, j + 1 );
253. }
254. }
255. }

* Funs.h

1. #ifndef FUNS\_H\_INCLUDED
2. #define FUNS\_H\_INCLUDED
4. **void** printOneDay( **int** year, **int** daySeqOfYear , **int** formatType );
5. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow );
7. #endif // FUNS\_H\_INCLUDED

* Dates.h

1. #ifndef DATE\_H\_INCLUDED
2. #define DATE\_H\_INCLUDED
4. #define YEAR\_MIN 2019
5. #define YEAR\_MAX 2026
6. #define YEAR\_NUM 8
7. #define DATE\_INFO\_BRIEF 0
8. #define DATE\_INFO\_FULL 1
10. **int** isLeapYear( **int** year );
11. **int** getDaySeqOnJan1( **int** year );
13. **int** getDaySeq( **int** year, **int** month, **int** day );
14. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day );
16. **int** getMonth( **int** year, **int** daySeqOfYear );
17. **int** getDay( **int** year, **int** daySeqOfYear );
18. **int** getDaySeqOfWeek( **int** year,**int** daySeqOfYear );
20. **int** getNextMonday( **int** year, **int** daySeqOfYear );
21. **int** getThisMonday ( **int** year, **int** day );
22. **int** getThisSunday( **int** year, **int** daySeqOfYear );
24. **void** setYearArray( **int** Years[], **int** yearnum, **int** year );
26. **void** initialDays(**int** Years[], **int** Days[][366][4], **int** yearNum );
28. #endif // DATE\_H\_INCLUDED
29. 实验三

* main.c

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "date.h"
4. #include "funs.h"
6. **const** **int** Years[5] = {2024,2025,2026,2027,2028,};
8. **const** **int** Month\_NORMAL\_YEAR [12] =
9. {31,28,31,30,31,30,31,31,30,31,30,31};
11. **const** **int** Month\_LEAP\_YEAR [12] =
12. {31,29,31,30,31,30,31,31,30,31,30,31};
14. **int** main()
15. {
17. **int** inputMonth,inputDay;
18. **int** i = 0;
19. **int** j = 0;
21. Day birthDay = {0};
22. Day prepareDay = {0};
23. Day printDay = {0};
24. Day prepareday = {0};
26. printf("please input month: \n");
27. scanf("%d", &inputMonth) ;
28. printf("\nplease input day: \n");
29. scanf("%d", &inputDay) ;
30. printf("\n");
32. **if** ( inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > Month\_LEAP\_YEAR [inputMonth - 1])
33. {
34. printf("Sorry, the input month and day are invalid.\n");
35. **return** 0;
36. }
38. **for** (i = 0; i < YEAR\_NUM; i++)
39. {
41. birthDay = setDay (Years[i],inputMonth,inputDay);
42. prepareDay = getDayBefore (birthDay, printDayRange);
43. prepareday = prepareDay;
45. printf("\%s%s%d\n", " ", "Birthday in Year ", Years[i]);
47. **if** (isDay(Years[i], inputMonth, inputDay)!= 1)
48. {
49. printf(" Not found.\n\n");
50. **continue**;
51. }
53. printDay = setDay (Years[i],inputMonth,inputDay + i);
55. printoneWeek(prepareday);
57. printf("\n\n");
58. }
60. **return** 0 ;
61. }

* Funs.c

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. **extern** **int** Month\_NORMAL\_YEAR [12] ;
9. **extern** **int** Month\_LEAP\_YEAR [12] ;
11. **void** printoneWeek( Day prepareday ){
12. Day remain = prepareday;
13. **int** i = 0;
14. **int** index = 1;
15. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
16. **if**(remain.dayseq + 7 - remain.weekDay > isLeapYear(prepareday.year) + 365){
17. prepareday.weekseq = 1;
18. }
19. printf("[%02d]",prepareday.weekseq);
20. **if**(i == 0)
21. {
22. **int** j;
23. **for**( j = 1; j < remain.weekDay ; j++){
24. printf("%10s"," ");
25. }
26. }
27. **int** k;
28. **for**( k = 0; k <= printDayRange;k ++){
29. **if**(k  == 0){
30. printf("  %02d.%02d.%02d",prepareday.year-100\*(prepareday.year/100), prepareday.month,prepareday.day);
31. prepareday.day ++;
32. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
34. **continue**;
35. }
36. **if**(k + remain.weekDay > 7 \*index){
37. getWeekSeqOfYear( prepareday.year, prepareday.month,getThisSunday( prepareday.year,prepareday.dayseq ));
38. **if**(prepareday.dayseq +7 > isLeapYear(prepareday.year) + 365){
39. prepareday.weekseq = 1;
40. }
41. printf("\n");
42. printf("[%02d]",prepareday.weekseq);
43. index ++;
44. }
45. **if**(k == printDayRange){
46. **if**(remain.dayseq + k > isLeapYear(prepareday.year) + 365){
47. prepareday.month = 1;
48. prepareday.day = 1;
49. prepareday.year ++;
50. prepareday.dayseq = 1;
51. prepareday.weekseq = 0;
52. printf(" %02d.%02d.%02d\*",prepareday.year-100\*(prepareday.year/100) , prepareday.month,prepareday.day);
53. prepareday.day ++;
54. prepareday.dayseq ++;
55. remain.dayseq = 2;
56. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
58. **continue**;
59. }**else** **if**(prepareday.month == 2 &&  isLeapYear(prepareday.year) == 1 && prepareday.day == 29){
61. printf("     %02d.%02d\*",prepareday.month,prepareday.day);
62. prepareday.month++;
63. prepareday.day = 1;
64. prepareday.dayseq ++;
65. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
67. **continue**;
68. }**else** **if**(prepareday.day >  Month\_NORMAL\_YEAR[prepareday.month - 1]){
69. prepareday.day = 1;
70. prepareday.month ++;
71. printf("    %02d.%02d\*",prepareday.month,prepareday.day);
72. prepareday.day++;
73. prepareday.dayseq ++;
74. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
76. **continue**;
77. }**else**{
78. printf("     %02d,%02d\*",prepareday.month,prepareday.day);
79. prepareday.day ++;
80. prepareday.dayseq ++;
81. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
83. **continue**;
84. }
86. }
87. **if**(remain.dayseq + k > isLeapYear(prepareday.year) + 365){
88. prepareday.month = 1;
89. prepareday.day = 1;
90. prepareday.year ++;
91. prepareday.dayseq = 1;
92. prepareday.weekseq = 0;
93. printf("  %02d.%02d.%02d",prepareday.year-100\*(prepareday.year/100) , prepareday.month,prepareday.day);
94. prepareday.day++;
95. prepareday.dayseq ++;
96. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
97. remain.dayseq = 2;
98. **continue**;
99. }**else** **if**(prepareday.month == 2 &&  isLeapYear(prepareday.year) == 1 && prepareday.day == 29){
101. printf("        %02d",prepareday.day);
102. prepareday.month++;
103. prepareday.day = 1;
104. prepareday.dayseq ++;
105. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
107. **continue**;
108. }**else** **if**(prepareday.day >  Month\_NORMAL\_YEAR[prepareday.month - 1]){
109. prepareday.day = 1;
110. prepareday.month ++;
111. printf("     %02d.%02d",prepareday.month,prepareday.day);
112. prepareday.day++;
113. prepareday.dayseq ++;
114. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
116. **continue**;
117. }**else**{
119. printf("        %02d",prepareday.day);
120. prepareday.dayseq += 1;
121. prepareday.day ++;
122. prepareday.weekseq = getWeekSeqOfYear( prepareday.year, prepareday.month, prepareday.day );
123. **if**(prepareday.dayseq > isLeapYear(remain.year) + 365){
124. prepareday.weekseq = 1;
125. }
126. **continue**;
127. }
129. }
130. }

* Dates.c

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. #define MONTH\_NUM 12
9. **int** getDay( **int** year, **int** daySeqOfYear )
10. {
11. **extern** **int**  Month\_LEAP\_YEAR[12];
12. **extern** **int**  Month\_NORMAL\_YEAR[12];
14. **int** i = 0 ;
15. **int** ret = daySeqOfYear ;
17. **if**( isLeapYear( year ) == 1 )
18. {
19. **while**( ret > Month\_LEAP\_YEAR[ i ] )
20. {
21. ret -= Month\_LEAP\_YEAR[ i ] ;
23. i ++ ;
24. }
25. }
27. **else**
28. {
29. **while**( ret > Month\_NORMAL\_YEAR[ i ] )
30. {
31. ret -= Month\_NORMAL\_YEAR[ i ] ;
33. i ++ ;
34. }
35. }
37. **return**  ret ;
38. }
40. **int** getDaySeq( **int** year, **int** month, **int** day )
41. {
42. **switch**( month )
43. {
44. **case** 12:
45. day += 30 ;
47. **case** 11:
48. day += 31 ;
50. **case** 10:
51. day += 30 ;
53. **case** 9:
54. day += 31 ;
56. **case** 8:
57. day += 31 ;
59. **case** 7:
60. day += 30 ;
62. **case** 6:
63. day += 31 ;
65. **case** 5:
66. day += 30 ;
68. **case** 4:
69. day += 31 ;
71. **case** 3:
72. day += 28 + isLeapYear( year ) ;
74. **case** 2:
75. day += 31 ;
77. **break** ;
78. }
80. **return** day ;
81. }
83. **int** getDaySeqOfWeek( **int** year, **int** daySeqOfYear )
84. {
86. daySeqOfYear += getDaySeqOnJan1( year ) - 1 ;
88. daySeqOfYear = daySeqOfYear % 7 ;
90. **return** daySeqOfYear ;
91. }
93. **int** getDaySeqOnJan1( **int** year )
94. {
95. **int** result ;
97. result = ( year - 1 +( year - 1) / 4 - ( year - 1)/ 100 +( year - 1)/ 400) % 7 + 1 ;
99. **return** result ;
100. }
102. **int** getMonth( **int** year, **int** dayseq )
103. {
104. **extern** **int**  Month\_LEAP\_YEAR[12];
105. **extern** **int**  Month\_NORMAL\_YEAR[12];
107. **int** temp = 1 ;
109. **if**( isLeapYear( year ) == 1 )
110. {
111. **while**( dayseq > Month\_LEAP\_YEAR[ temp - 1 ] )
112. {
113. dayseq -= Month\_LEAP\_YEAR[ temp - 1 ] ;
115. temp ++ ;
116. }
117. }
119. **else**
120. {
121. **while**( dayseq > Month\_NORMAL\_YEAR[ temp - 1 ] )
122. {
123. dayseq -= Month\_NORMAL\_YEAR[ temp - 1 ] ;
125. temp ++ ;
126. }
127. }
128. **return** temp ;
129. }
131. **int** getNextMonday( **int** year, **int** day )
132. {
133. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 1)
134. {
135. **if**(( day + getDaySeqOnJan1( year ) - 1 ) % 7 == 0)
136. {
137. day ++ ;
138. }
140. **else**
141. {
142. day += (7 -( day + getDaySeqOnJan1( year ) - 1 ) % 7) + 1 ;
143. }
144. }
146. **return** day ;
147. }
149. **int** getThisMonday ( **int** year, **int** day )
150. {
151. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 1)
152. {
153. **if**(( day + getDaySeqOnJan1( year ) - 1 ) % 7 == 0)
154. {
155. day -= 6 ;
156. }
158. **else**
159. {
160. day -= ( day + getDaySeqOnJan1( year ) - 1 ) % 7 - 1 ;
161. }
162. }
164. **return** day ;
165. }
167. **int** getThisSunday( **int** year, **int** day )
168. {
169. **if**( ( day + getDaySeqOnJan1( year ) - 1 ) % 7 != 0)
170. {
171. day += 7 -(day + getDaySeqOnJan1( year ) - 1 ) % 7 ;
172. }
174. **return** day ;
175. }
177. **int** isLeapYear( **int** year )
178. {
179. **if**( ( year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )
180. {
181. **return** 1 ;
182. }
184. **return** 0 ;
185. }
187. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day )
188. {
189. **int** week ;
191. **int** daySeqOfYear = getDaySeq( year, month, day ) ;
193. **int** currentyear = 365 + isLeapYear( year ) ;
195. **if**( daySeqOfYear > currentyear )
196. {
197. **if**( getDaySeqOfWeek( year + 1, daySeqOfYear - currentyear) == 0)
198. {
199. week = ( daySeqOfYear + getDaySeqOnJan1( year ) ) / 7 ;
200. }
202. **else**
203. {
204. week = ( daySeqOfYear + getDaySeqOnJan1( year ) +
205. ( 7 - getDaySeqOfWeek( year + 1, daySeqOfYear - currentyear))) / 7 ;
206. }
207. }
209. **else**
210. {
211. **if**( getDaySeqOfWeek( year , daySeqOfYear ) == 0)
212. {
213. week = ( daySeqOfYear + getDaySeqOnJan1( year ) ) / 7 ;
214. }
216. **else**
217. {
218. week = ( daySeqOfYear + getDaySeqOnJan1( year ) +
219. ( 7 - getDaySeqOfWeek( year , daySeqOfYear)))/ 7 ;
220. }
221. }
223. **return** week;
224. }
226. **void** setYearArray( **int** Years[], **int** yearnum, **int** year )
227. {
228. **int** num1 ;
230. **for**( num1 = 0; num1 < yearnum ; num1 ++ , year ++)
231. {
232. Years[ num1 ] = year;
233. }
234. }

237. **int** isDay(**int** year, **int** month, **int** day){
239. **extern** **int** Month\_NORMAL\_YEAR [12] ;
241. **if**( isLeapYear(year) != 1 && month == 2 && day >= 29)
242. {
243. **return** -1;
244. }
245. **if**(month < 1 || month > 12){
246. **return** -1;
247. }
248. **if**(isLeapYear(year) == 1 && month == 2 && day == 29){
249. **return** 1;
250. }
251. **if**(day > Month\_NORMAL\_YEAR [month-1] || day < 1){
252. **return** -1;
253. }
254. **return** 1;
255. }
256. Day setDay(**int** year, **int** month, **int** day){
257. Day ret ;
258. ret.day = day ;
259. ret.year = year;
260. ret.month = month;
261. ret.dayseq = getDaySeq( year,  month,  day );
262. ret.weekseq = getWeekSeqOfYear( year, month, day );
263. ret.weekDay = getweekDay( year, ret.dayseq);
264. **return** ret;
265. }
267. Day getDayBefore( Day currentDay,**int** interval ){
268. Day ret ;
269. **if**(currentDay.month == 1 && currentDay.day - interval <= 0 ){
270. ret.day = currentDay.day+31 - interval;
271. ret.month = 12;
272. ret.dayseq = isLeapYear(currentDay.year -1) + 365 + currentDay.dayseq - interval;
273. ret.weekDay = currentDay.weekDay - interval <= 0 ? currentDay.weekDay - interval+7 : currentDay.weekDay - interval;
274. ret.weekseq = getWeekSeqOfYear(  currentDay.year - 1, ret.month, ret.day );
275. ret.year = currentDay.year - 1;
276. **return** ret;
277. }
278. ret.month = currentDay.day - interval > 0 ?currentDay.month :currentDay.month - 1;
279. ret.year = currentDay.year;
280. ret.dayseq = currentDay.dayseq - interval;
281. ret.day = getDay(  ret.year, ret.dayseq );
282. ret.weekDay = getweekDay(ret.year,ret.dayseq);
283. ret.weekseq = getWeekSeqOfYear(  ret.year, ret.month, ret.day );

286. **return** ret;
287. }
289. **int** getweekDay(**int** year,**int** daySeq){
290. **int** ret = 0;
291. ret = daySeq+7-getThisSunday(year , daySeq);
292. **return** ret ;
293. }
295. Day getDayAfter( Day currentDay,**int** interval );
296. **int** getTwoDaysInterval( Day startDay, Day endDay);
297. **void** printDay( Day currentDay, **int** displayFormat );

* Funs.h

1. #ifndef FUNS\_H\_INCLUDED
2. #define FUNS\_H\_INCLUDED
4. **void** printoneWeek( Day day);
6. #endif // FUNS\_H\_INCLUDED

* Dates.h

1. #ifndef DATE\_H\_INCLUDED
2. #define DATE\_H\_INCLUDED
4. #define YEAR\_MIN 2000
5. #define YEAR\_MAX 2030
6. #define YEAR\_NUM 5
7. #define DATE\_INFO\_BRIEF 0
8. #define DATE\_INFO\_FULL 1
9. #define YEAR\_NUM 5
10. #define printDayRange 3
12. **typedef** **struct** day{
13. **int** year;
14. **int** dayseq;
15. **int** month;
16. **int** day;
17. **int** weekseq;
18. **int** weekDay;
19. } Day;
20. **int** isLeapYear( **int** year );
21. **int** getDaySeqOnJan1( **int** year );
23. **int** getDaySeq( **int** year, **int** month, **int** day );
24. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day );
26. **int** getMonth( **int** year, **int** daySeqOfYear );
27. **int** getDay( **int** year, **int** daySeqOfYear );
28. **int** getDaySeqOfWeek( **int** year,**int** daySeqOfYear );
30. **int** getNextMonday( **int** year, **int** daySeqOfYear );
31. **int** getThisMonday ( **int** year, **int** day );
32. **int** getThisSunday( **int** year, **int** daySeqOfYear );
34. **void** setYearArray( **int** Years[], **int** yearnum, **int** year );
36. **int** getweekDay(**int** year,**int** daySeq);
37. **int** isDay(**int** year, **int** month, **int** day);
38. Day setDay(**int** year, **int** month, **int** day);
39. Day getDayBefore( Day currentDay,**int** interval );
40. Day getDayAfter( Day currentDay,**int** interval );
41. **int** getTwoDaysInterval( Day startDay, Day endDay);
42. **void** printDay( Day currentDay, **int** displayFormat );
44. #endif // DATE\_H\_INCLUDED