

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 | 曹源 | 学号 | U202412268 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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

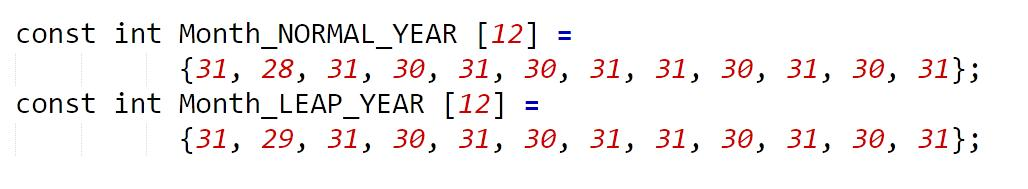
# 实验一（Calendar08. **不同年份秋季校历首周对比**）

## 试验任务

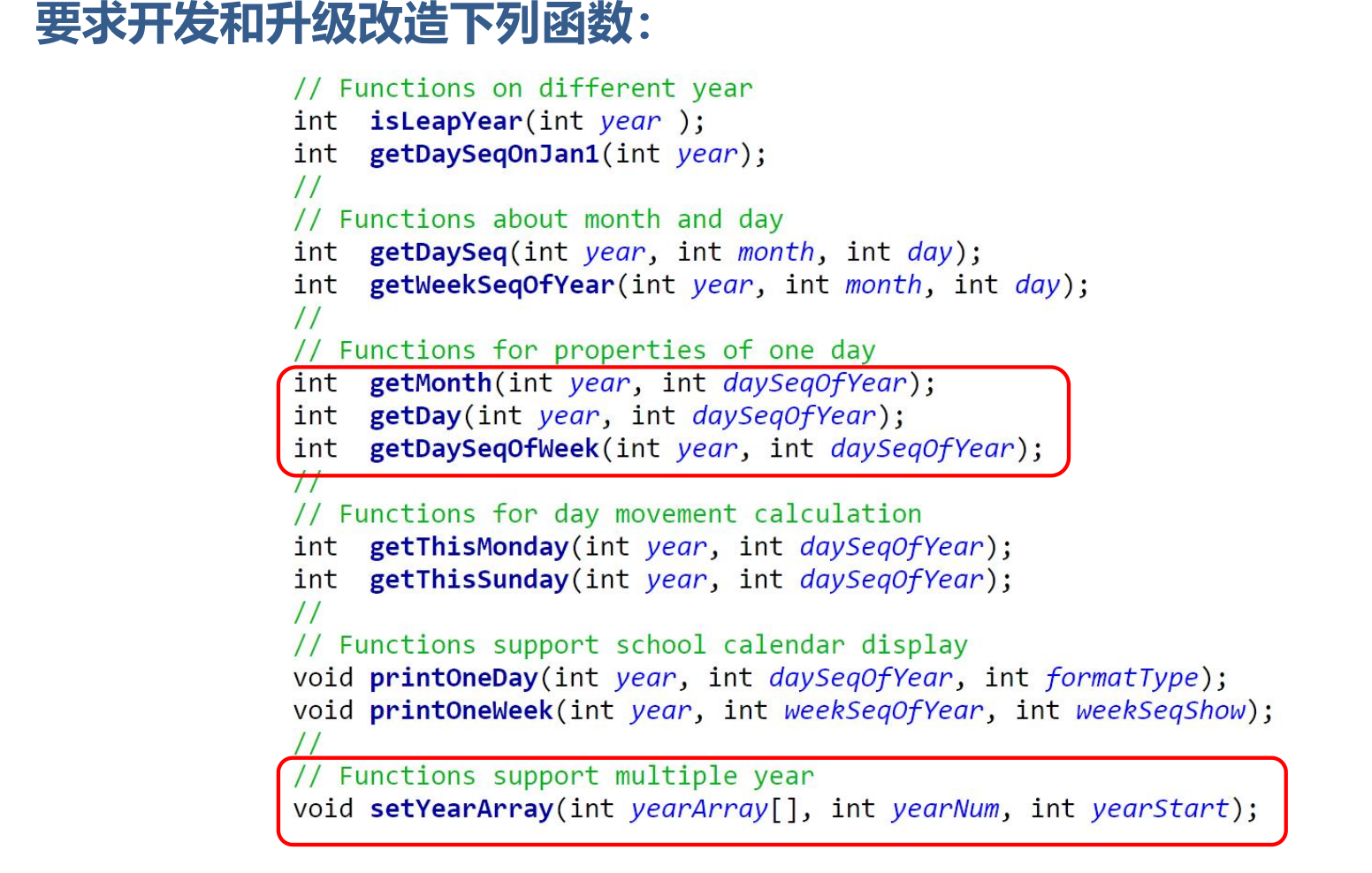
–用数组改写打印华中科技大学校历的程序，具备打印校历的能力，比较多个年份的校历秋季校历的首日，寻找开学最早的那一年

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

–用两个只读的全局的一维数组记录不同月份的天数长度–基于数组改进现有的日历程序，改进计算日期的函数

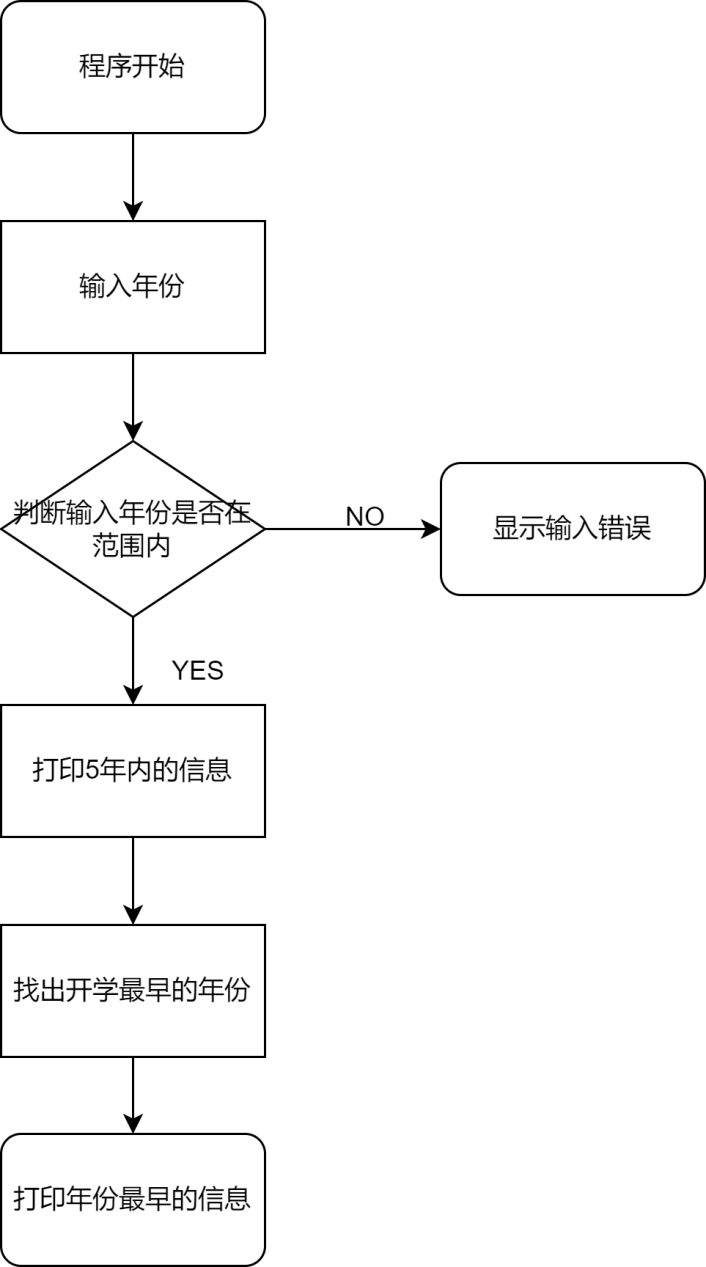






## 实验步骤

实验思路



重要代码

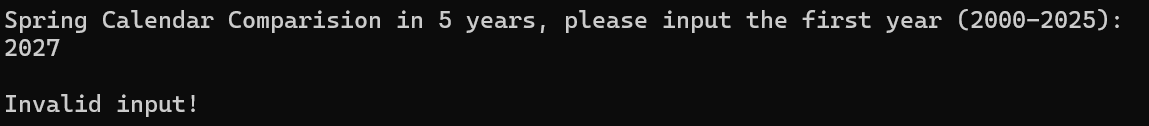
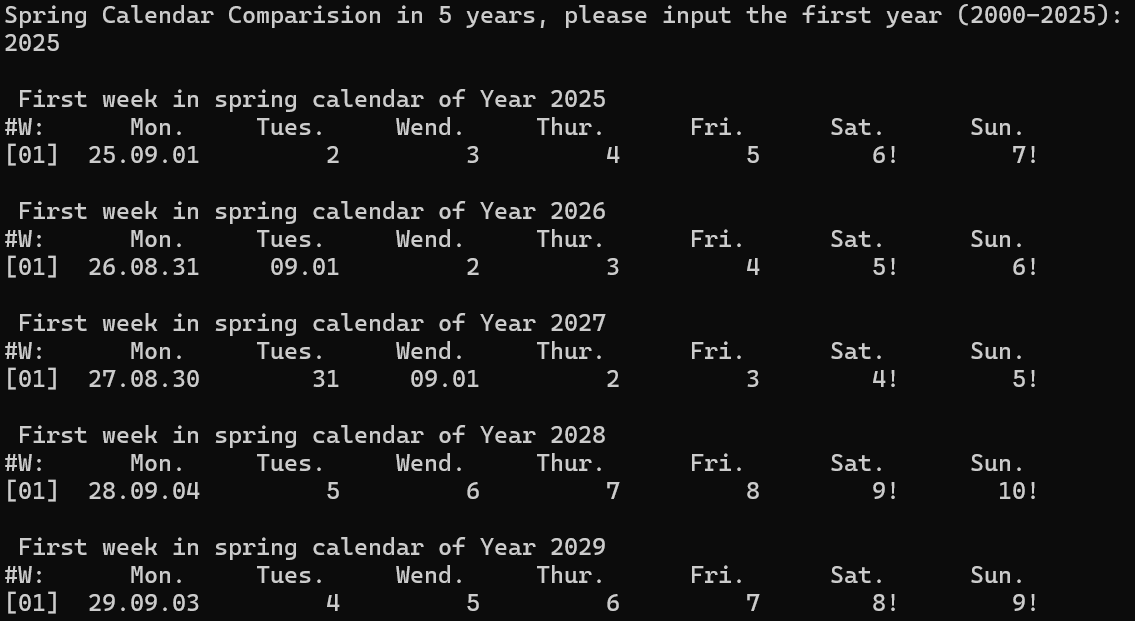
printoneWeek(int year, int weekSeqOfYear, int weekSeqShow )

getWeekSeqOfYear( int year, int month, int day )

## 代码测试

分别输入2013，2025和2027的数据





2013和2025未超过有效区间，2027超过限制，而程序运行也符合预测结果，程序运行正确。

## 实验结论

代码达到实验目标。

## 实验总结

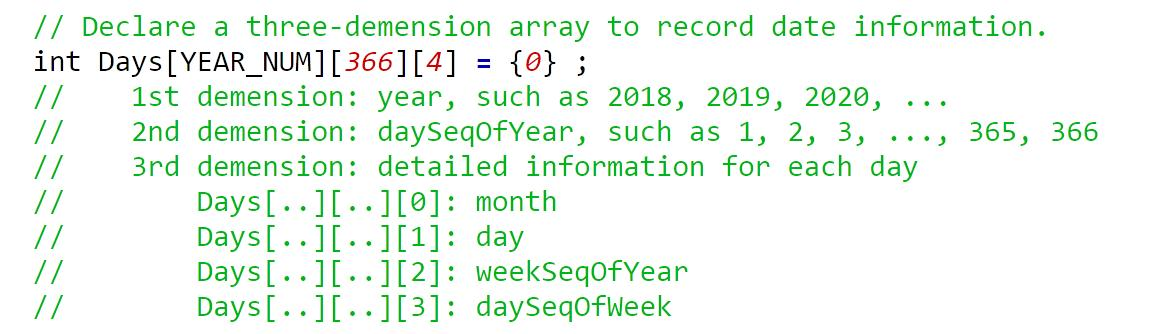
程序基本符合要求，但在部分细节方面有一些问题。包括但不仅限于对scanf输入并非整数的反应，数组索引的越界问题和闰年的规则考虑问题，代码的可维护性不高的问题。这些问题仍需练习改正。

# 实验二（**Calendar09. 寻找生日（多维数组））**

## 试验任务

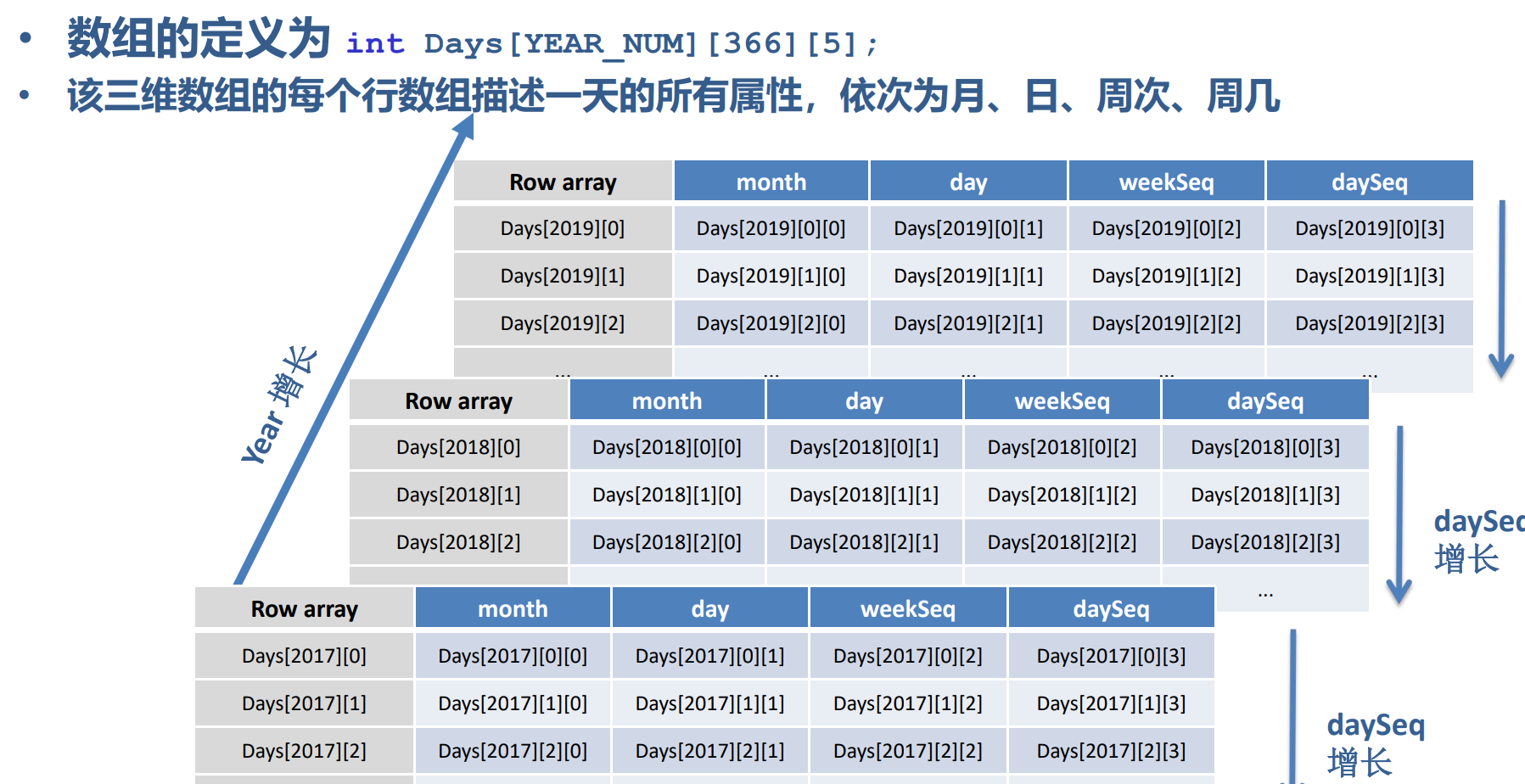
–用三维数组记录多年的日期数据，查找某人的生日，并打印该生日所在周的周历，计算其在周末过生日的次数

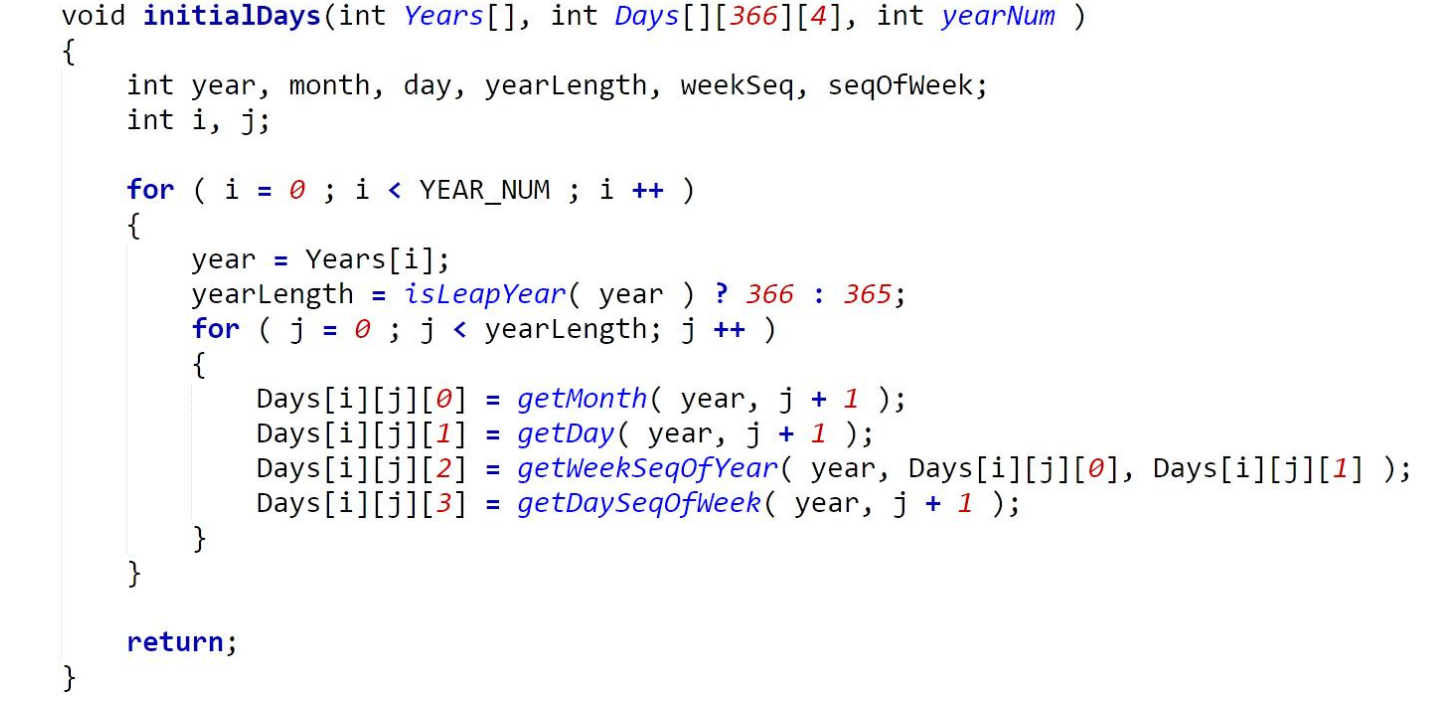
–三维数组的定义如下



•数组的定义为 int Days[YEAR\_NUM][366][5];

•该三维数组的每个行数组描述一天的所有属性，依次为月、日、周次、周几

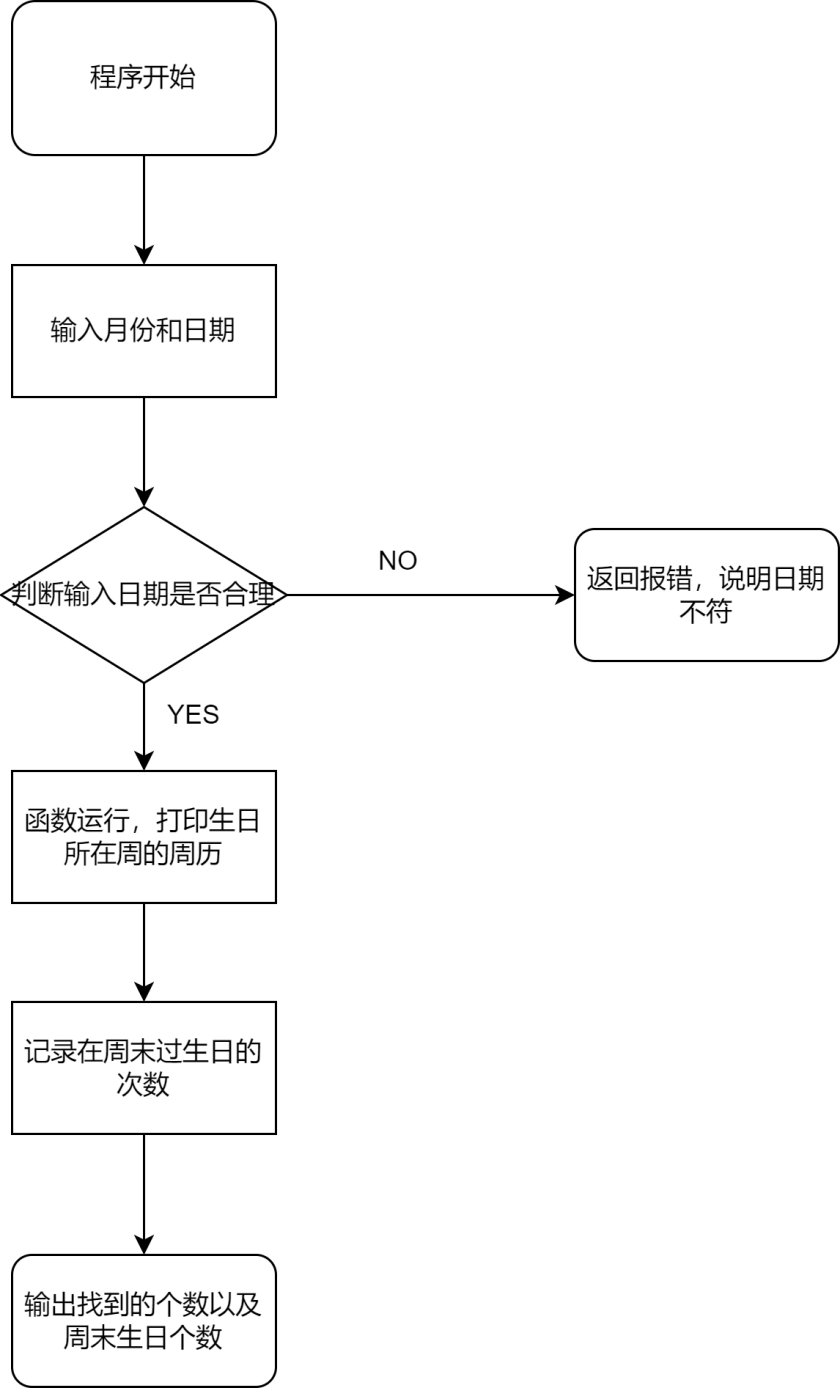






## 实验步骤

编程思路



重要代码

initialDays(int Years[], int Days[][366][4], int yearNum )

getNextMonday( int year, int daySeqOfYear )

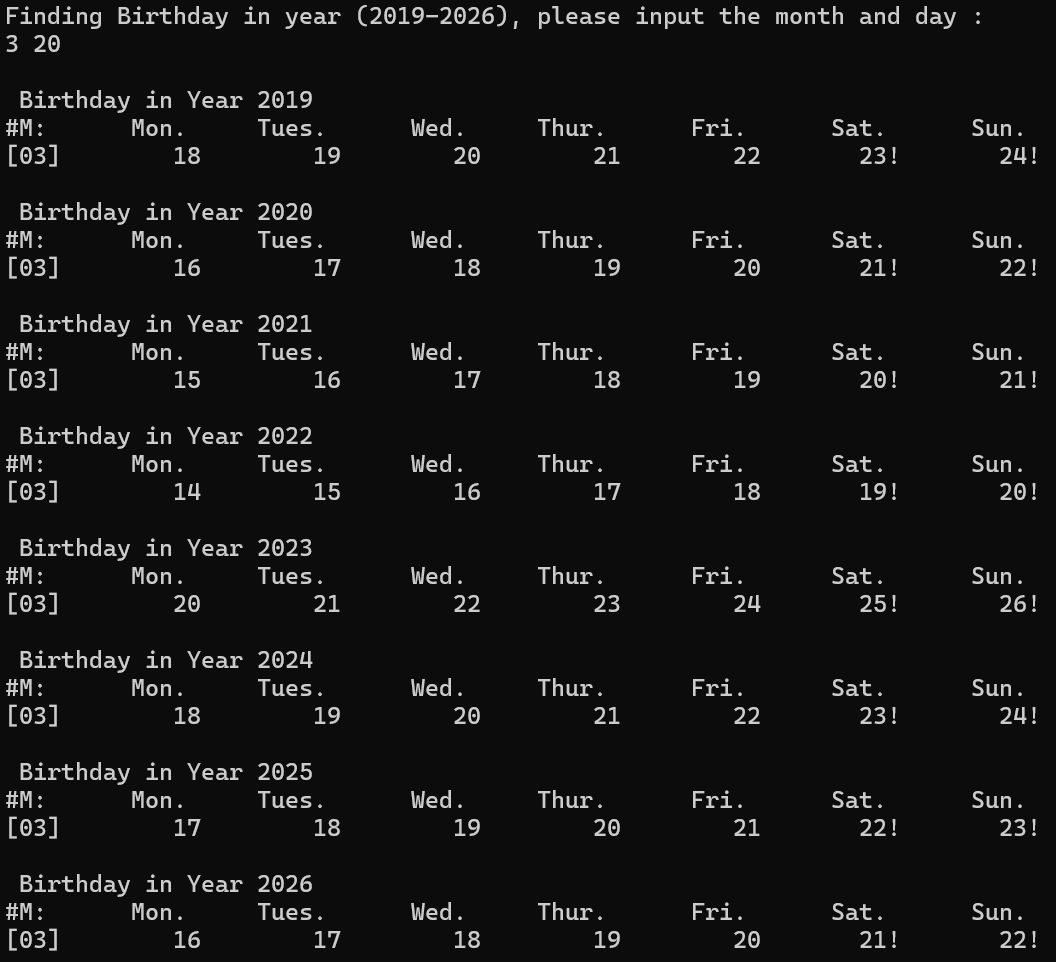
getThisMonday ( int year, int day )

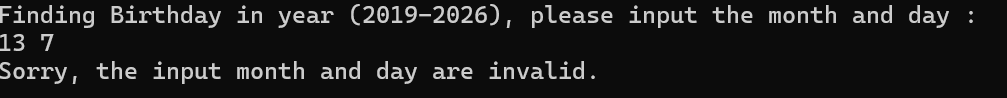
getThisSunday( int year, int daySeqOfYear )

## 代码测试

输入代表性例子，寻找12月6日，3月20日和13月7日的生日







程序运行符合条件，能够查找生日的日期查出来了，不能查找生日的日期返回报错。

## 实验结论

代码达到实验目标。

## 实验总结

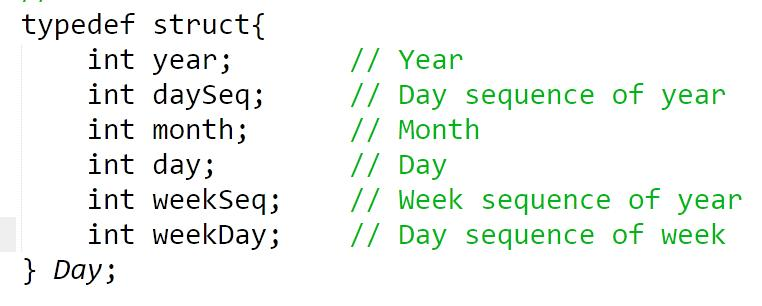
对inputday和inputmonth的验证开始较不完善，对于一些越界日期没有做到应有的判断，后来在实验过程中纠正了回来。开始存在程序结构的问题，分行难以阅读，后来予以改正，使程序更具有可读性。

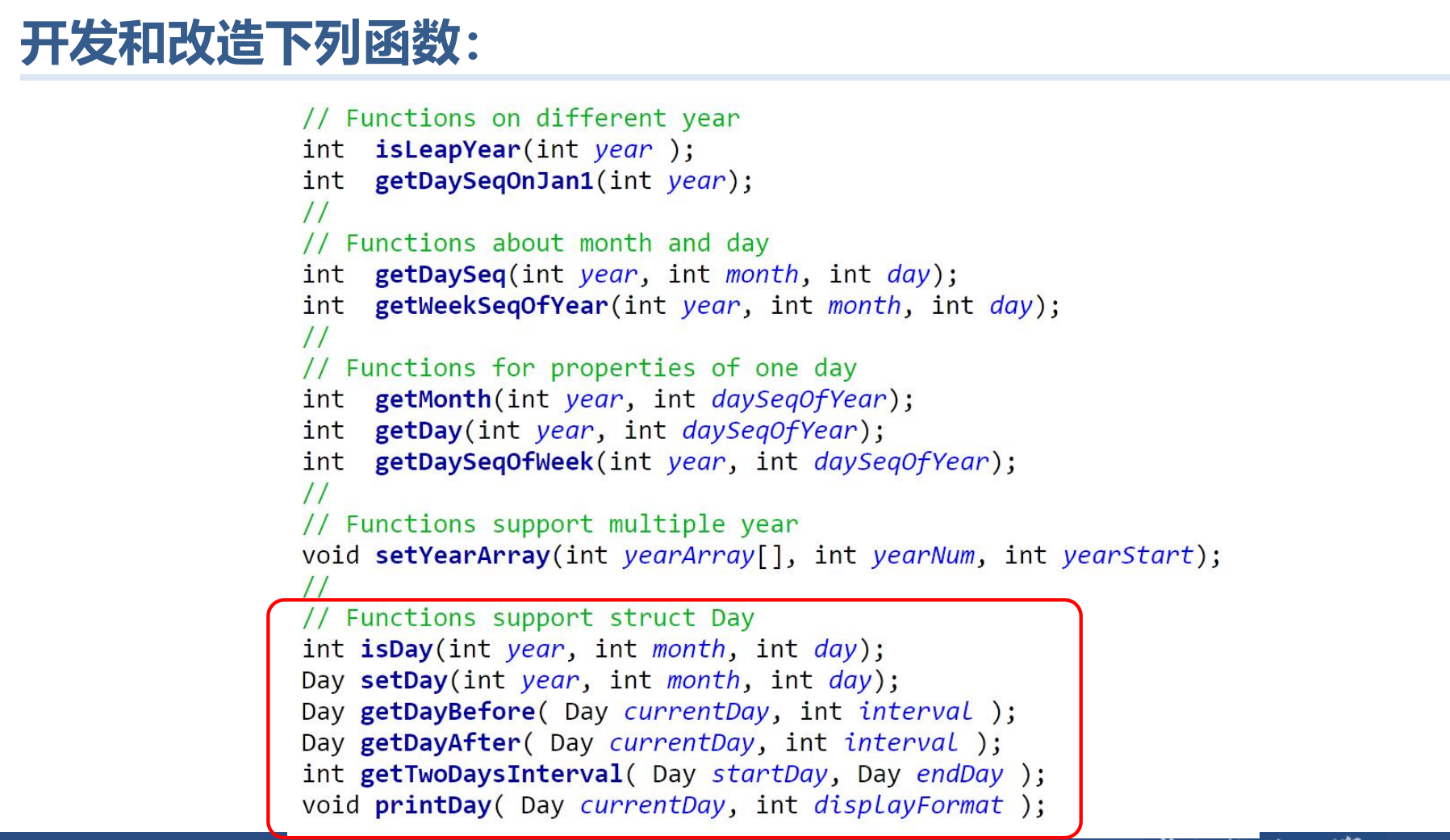
# 实验三（**Calendar09. 寻找生日（结构体））**

## 试验任务

–假定某生日趴需要三天时间准备，输入某人的生日，通过日期偏移计算获得前三天并打印相关周历

–用日期结构体记录单一结构体的所有属性







## 实验步骤

实验思路



重要代码

typedef struct day{

int year;

int dayseq;

int month;

int day;

int weekseq;

int weekDay;

} Day;

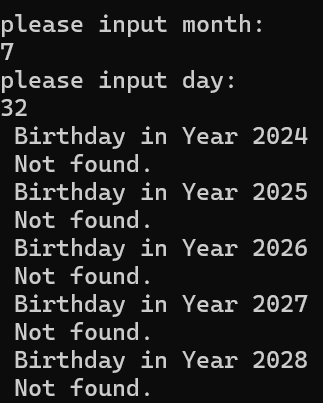
getDay( int year, int daySeqOfYear );

## 代码测试

分别选取几个典型日期进行计算打印。如5月1日，3月12日和7月32日.







无论是合理日期还是不合理日期，程序结果都符合预期。

## 实验结论

代码达到实验目标。

## 实验总结

实验函数的模块化使得程序更加简洁明了，但是程序在分块函数中的数组越界问题和内存占用没有达到最优。程序交互性还有所不足，比如没有让用户知道输入的格式，以及输出格式的用意。总之，程序仍需改进，但是模块化函数确实使程序更加思路清晰。

# 本课程学习总结

学习c语言的历程和总结

C语言的世界无疑是浩瀚而广袤的，而初入c语言世界，我就遭遇了许多困难。从scanf函数的变量忘记加上取地址符号，到数组的越界乱码问题，再到递归函数的逻辑错误，甚至是最基本的大括号数量错误，一路上遭遇了许多拦路虎......但是随着经验的增长，我慢慢看到了c语言最深处的魅力：精确的表述和严谨的逻辑，每一次深深反思的绞尽脑汁和突破困难的快感，优化自身程序使代码更加简洁的精益求精，都让我深深地体会到c语言创造的快感。学c语言只学了半年，但蓦然回首，却发现一路受益良多，日渐精进。

程序中出现的问题以及改进

代码的规范性是开始最大的问题。缩进，空格，空行，注释......开始编程的我对这些细微之处毫不在意，以为没有这些程序依然浅显易懂，但到后来程序的日渐复杂，如果不对这些细微之处更加严格要求，我甚至看不懂自己写的是什么。从此我就知道，要对代码的规范性和可读性有更高的要求。不仅如此，在编写日历过程中，数组越界和指针指向错误也是大问题，轻则出现乱码，重则程序重写，指针和数组的运用是c语言的重要点，也是难度点。尤其是指针动态分配内存后未释放内存，更是个隐患，有时候不知道为什么就会让运行程序崩溃。更别说日历本身带有的平年闰年，日期更替等类似细微之处。c语言的细微之处太多太多，需要仔细注意。

代码规范与调试技巧

养成注释的良好习惯，让别人能够清晰地看到你的编程思路，而不是看到天书一样懵。一行只放一句代码，不要挤在一块。If和for养成用大括号将内部代码括起来的习惯，不要只有一句话就不括起来。把函数以模块的方式括起来，这样能够更好地重复应用，使程序更好维护。调试的时候注意细微部分，模块化调试可以快速找到问题所在处。

学习记录和心得

编写代码，寻找思路的时候是痛苦的，绞尽脑汁颅内模拟却又想不出来的时候真的很难受，但是当想出来的时候就如同雨过天晴一样让人极为舒畅。当没有思路想要放弃的时候，不妨坚持一下，我相信最后一定会成功的。所以，让我们一起享受c语言绞尽脑汁的过程！

# 附录

# 实验一（Calendar08. 不同年份秋季校历首周对比）

1. main.c
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. // 平年每月的天数
8. **const** **int** Month\_NORMAL\_YEAR[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
9. // 闰年每月的天数
10. **const** **int** Month\_LEAP\_YEAR[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
12. **int** main()
13. {
14. **int** year, Years[YEAR\_NUM];
15. printf("Spring Calendar Comparision in %d years, please input the first year (%d-%d): \n", YEAR\_NUM, YEAR\_MIN, YEAR\_MAX - YEAR\_NUM);
16. scanf("%d", &year);
18. **if** (year < YEAR\_MIN || year + YEAR\_NUM > YEAR\_MAX) {
19. printf("\nInvalid input!\n");
20. **return** 0;
21. }
22. setYearArray(Years, YEAR\_NUM, year);
24. **int** i;
25. **int** startDays[YEAR\_NUM] = {0};
27. **for** (i = 0; i < YEAR\_NUM; i++) {
28. // 打印每年春季学期第一周的信息
29. printf("\n%s%s%d\n", " ", "First week in spring calendar of Year ", Years[i]);
30. printf("#W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wend.", "Thur.", "Fri.", "Sat.", "Sun.");
31. printoneWeek(Years[i], getWeekSeqOfYear(Years[i], 9, 4), 1);
32. startDays[i] = getThisMonday(Years[i], getWeekSeqOfYear(Years[i], 9, 4));
33. }
34. **int** min = 0;
35. **for** (i = 1; i < YEAR\_NUM; i++) {
36. min = (startDays[i] < startDays[min])? i : min;
37. }
39. printf("\n%s%s%d\n", " ", "Earliest Spring Semester is in Year ", Years[min]);
41. **return** 0;
42. funs.h
43. #ifndef FUNS\_H\_INCLUDED
44. #define FUNS\_H\_INCLUDED
46. **void** printOneDay( **int** year, **int** daySeqOfYear , **int** formatType );
47. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow );
49. #endif // FUNS\_H\_INCLUDED
50. funs.c
51. #include <stdio.h>
52. #include <stdlib.h>
53. #include "date.h"
54. #include "funs.h"
56. **void** printOneDay(**int** year, **int** daySeqOfYear, **int** formatType)
57. {
58. **int** day = getDay(year, daySeqOfYear);
59. **int** month = getMonth(year, daySeqOfYear);
60. **int** week = getDaySeqOfWeek(year, daySeqOfYear);
62. **if** (formatType == 1) {
63. printf("%2s%02d.%02d.%02d", " ", year % 100, month, day);
64. } **else** **if** (day == 1) {
65. **if** (month == 1) {
66. **if** (week == 6 || week == 0) {
67. printf("%1s%02d.%02d.%02d!", " ", year % 100, month, day);
68. } **else** {
69. printf("%2s%02d.%02d.%02d", " ", year % 100, month, day);
70. }
71. } **else** **if** (week == 6 || week == 0) {
72. printf("%4s%02d.%02d!", " ", month, day);
73. } **else** {
74. printf("%5s%02d.%02d", " ", month, day);
75. }
76. } **else** {
77. **if** (week == 6 || week == 0) {
78. printf("%9d!", day);
79. } **else** {
80. printf("%10d", day);
81. }
82. }
83. }
85. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow)
86. {
87. printf("[%02d]", weekSeqShow);
89. **int** sStartSeqOfYear = 7 \* (weekSeqOfYear - 1) - getDaySeqOnJan1(year) + 2;
90. **int** sEndSeqOfYear = sStartSeqOfYear + 6;
91. **int** currentyear = 365 + isLeapYear(year);
92. **int** daySeqOfYear = sStartSeqOfYear;
94. **if** (weekSeqShow == 1) {
95. printOneDay(year, daySeqOfYear, DATE\_INFO\_FULL);
96. daySeqOfYear++;
97. }
99. **for** (; daySeqOfYear <= sEndSeqOfYear; daySeqOfYear++) {
100. **if** (daySeqOfYear <= currentyear) {
101. printOneDay(year, daySeqOfYear, DATE\_INFO\_BRIEF);
102. } **else** {
103. printOneDay(year + 1, daySeqOfYear - currentyear, DATE\_INFO\_BRIEF);
104. }
105. }
106. printf("\n");
107. }
108. date.h
109. #ifndef DATE\_H\_INCLUDED
110. #define DATE\_H\_INCLUDED
112. #define YEAR\_MIN 2000
113. #define YEAR\_MAX 2030
114. #define YEAR\_NUM 5
115. #define DATE\_INFO\_BRIEF 0
116. #define DATE\_INFO\_FULL 1
118. **int** isLeapYear( **int** year );
119. **int** getDaySeqOnJan1( **int** year );
121. /\*int getMonthLength( int year , int month );\*/
122. **int** getDaySeq( **int** year, **int** month, **int** day );
123. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day );
125. **int** getMonth( **int** year, **int** daySeqOfYear );
126. **int** getDay( **int** year, **int** daySeqOfYear );
127. **int** getDaySeqOfWeek( **int** year,**int** daySeqOfYear );
129. **int** getNextMonday( **int** year, **int** daySeqOfYear );
130. **int** getThisMonday ( **int** year, **int** day );
131. **int** getThisSunday( **int** year, **int** daySeqOfYear );
133. **void** setYearArray( **int** Years[], **int** yearnum, **int** year );
135. #endif // DATE\_H\_INCLUDED
136. date.c
137. #include <stdio.h>
138. #include <stdlib.h>
139. #include "date.h"
140. #include "funs.h"
142. #define MONTH\_NUM 12

145. **int** getDay(**int** year, **int** daySeqOfYear)
146. {
147. **extern** **int** Month\_LEAP\_YEAR[12];
148. **extern** **int** Month\_NORMAL\_YEAR[12];
149. **int** i = 0;
150. **int** ret = daySeqOfYear;
151. **if** (isLeapYear(year) == 1) {
152. **while** (ret > Month\_LEAP\_YEAR[i]) {
153. ret -= Month\_LEAP\_YEAR[i];
154. i++;
155. }
156. } **else** {
157. **while** (ret > Month\_NORMAL\_YEAR[i]) {
158. ret -= Month\_NORMAL\_YEAR[i];
159. i++;
160. }
161. }
162. **return** ret;
163. }

166. **int** getDaySeq(**int** year, **int** month, **int** day)
167. {
168. **switch** (month) {
169. **case** 12:
170. day += 30;
171. **case** 11:
172. day += 31;
173. **case** 10:
174. day += 30;
175. **case** 9:
176. day += 31;
177. **case** 8:
178. day += 31;
179. **case** 7:
180. day += 30;
181. **case** 6:
182. day += 31;
183. **case** 5:
184. day += 30;
185. **case** 4:
186. day += 31;
187. **case** 3:
188. day += 28 + isLeapYear(year);
189. **case** 2:
190. day += 31;
191. **break**;
192. }
193. **return** day;
194. }

197. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
198. {
199. daySeqOfYear += getDaySeqOnJan1(year) - 1;
200. daySeqOfYear = daySeqOfYear % 7;
201. **return** daySeqOfYear;
202. }

205. **int** getDaySeqOnJan1(**int** year)
206. {
207. **int** result;
208. result = (year - 1 + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400) % 7 + 1;
209. **return** result;
210. }

213. **int** getMonth(**int** year, **int** dayseq)
214. {
215. **extern** **int** Month\_LEAP\_YEAR[12];
216. **extern** **int** Month\_NORMAL\_YEAR[12];
217. **int** temp = 1;
218. **if** (isLeapYear(year) == 1) {
219. **while** (dayseq > Month\_LEAP\_YEAR[temp - 1]) {
220. dayseq -= Month\_LEAP\_YEAR[temp - 1];
221. temp++;
222. }
223. } **else** {
224. **while** (dayseq > Month\_NORMAL\_YEAR[temp - 1]) {
225. dayseq -= Month\_NORMAL\_YEAR[temp - 1];
226. temp++;
227. }
228. }
229. **return** temp;
230. }

233. **int** getNextMonday(**int** year, **int** day)
234. {
235. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 1) {
236. **if** ((day + getDaySeqOnJan1(year) - 1) % 7 == 0) {
237. day++;
238. } **else** {
239. day += (7 - (day + getDaySeqOnJan1(year) - 1) % 7) + 1;
240. }
241. }
242. **return** day;
243. }
245. **int** getThisMonday(**int** year, **int** day)
246. {
247. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 1) {
248. **if** ((day + getDaySeqOnJan1(year) - 1) % 7 == 0) {
249. day -= 6;
250. } **else** {
251. day -= ((day + getDaySeqOnJan1(year) - 1) % 7 - 1);
252. }
253. }
254. **return** day;
255. }

258. **int** getThisSunday(**int** year, **int** day)
259. {
260. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 0) {
261. day += 7 - ((day + getDaySeqOnJan1(year) - 1) % 7);
262. }
263. **return** day;
264. }

267. **int** isLeapYear(**int** year)
268. {
269. **if** ((year % 4 == 0 && year % 100!= 0) || year % 400 == 0) {
270. **return** 1;
271. }
272. **return** 0;
273. }

276. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day)
277. {
278. **int** week;
279. **int** daySeqOfYear = getDaySeq(year, month, day);
280. **int** currentyear = 365 + isLeapYear(year);
281. **if** (daySeqOfYear > currentyear) {
282. **if** (getDaySeqOfWeek(year + 1, daySeqOfYear - currentyear) == 0) {
283. week = (daySeqOfYear + getDaySeqOnJan1(year)) / 7;
284. } **else** {
285. week = (daySeqOfYear + getDaySeqOnJan1(year) + (7 - getDaySeqOfWeek(year + 1, daySeqOfYear - currentyear))) / 7;
286. }
287. } **else** {
288. **if** (getDaySeqOfWeek(year, daySeqOfYear) == 0) {
289. week = (daySeqOfYear + getDaySeqOnJan1(year)) / 7;
290. } **else** {
291. week = (daySeqOfYear + getDaySeqOnJan1(year) + (7 - getDaySeqOfWeek(year, daySeqOfYear))) / 7;
292. }
293. }
294. **return** week;
295. }

298. **void** setYearArray(**int** Years[], **int** yearnum, **int** year)
299. {
300. **int** num1;
301. **for** (num1 = 0; num1 < yearnum; num1++, year++) {
302. Years[num1] = year;
303. }

# 实验二 （Calendar09. 寻找生日（多维数组））

1. main.c
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. **const** **int** Month\_NORMAL\_YEAR[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
8. **const** **int** Month\_LEAP\_YEAR[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
10. **int** main()
11. {
12. // Declare variables
13. **int** inputMonth, inputDay, Years[YEAR\_NUM];
14. // Declare a three-dimension array to record date information.
15. **int** Days[YEAR\_NUM][366][4] = {0};
17. // Display the program information
18. printf("Finding Birthday in year (%d-%d), please input the month and day : \n", YEAR\_MIN, YEAR\_MAX);
19. scanf("%d %d", &inputMonth, &inputDay);
21. **if** (inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > 31) {
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 = 0;
30. **int** weekendNum = 0;
32. **int** i, j;
33. **for** (i = 0; i <= YEAR\_NUM - 1; i++) {
34. printf("\n%1s%s%d\n", "", "Birthday in Year ", Years[i]);
35. **for** (j = 0; j < 366; j++) {
36. **if** ((Days[i][j][0] == inputMonth) && (Days[i][j][1] == inputDay)) {
37. totalNum++;
38. printf("#M:%10s%10s%10s%10s%10s%10s%10s\n",
39. "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
40. printoneWeek(Years[i], getWeekSeqOfYear(Years[i], inputMonth, inputDay), inputMonth);
41. **if** ((Days[i][j][3] == 6) || (Days[i][j][3] == 0)) {
42. weekendNum++;
43. }
44. **break**;
45. }
46. **if** (j == 365) {
47. printf(" Not found.\n");
48. }
49. }
50. }
52. printf("\nTotal %d birthday are found, %d of them are in weekends.\n", totalNum, weekendNum);
53. **return** 0;
54. }
56. **void** initialDays(**int** Years[], **int** Days[][366][4], **int** yearNum)
57. {
58. **int** year, month, day, yearLength, weekSeq, seqOfWeek;
59. **int** i, j;
61. **for** (i = 0; i < YEAR\_NUM; i++) {
62. year = Years[i];
63. yearLength = isLeapYear(year)? 366 : 365;
64. **for** (j = 0; j < yearLength; j++) {
65. Days[i][j][0] = getMonth(year, j + 1);
66. Days[i][j][1] = getDay(year, j + 1);
67. Days[i][j][2] = getWeekSeqOfYear(year, Days[i][j][0], Days[i][j][1]);
68. Days[i][j][3] = getDaySeqOfWeek(year, j + 1);
69. }
70. }
71. **return**;
72. }
73. funs.h
74. #ifndef FUNS\_H\_INCLUDED
75. #define FUNS\_H\_INCLUDED
77. **void** printOneDay( **int** year, **int** daySeqOfYear , **int** formatType );
78. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow );
80. #endif // FUNS\_H\_INCLUDED
81. funs.c
82. #include <stdio.h>
83. #include <stdlib.h>
84. #include "date.h"
85. #include "funs.h"
87. **void** printOneDay(**int** year, **int** daySeqOfYear, **int** formatType)
88. {
89. **int** day = getDay(year, daySeqOfYear);
90. **int** month = getMonth(year, daySeqOfYear);
91. **int** week = getDaySeqOfWeek(year, daySeqOfYear);
93. **if** (formatType == 1) {
94. printf("%2s%02d.%02d.%02d", " ", year % 100, month, day);
95. } **else** **if** (day == 1) {
96. **if** (month == 1) {
97. **if** (week == 6 || week == 0) {
98. printf("%1s%02d.%02d.%02d!", " ", year % 100, month, day);
99. } **else** {
100. printf("%2s%02d.%02d.%02d", " ", year % 100, month, day);
101. }
102. } **else** **if** (week == 6 || week == 0) {
103. printf("%4s%02d.%02d!", " ", month, day);
104. } **else** {
105. printf("%5s%02d.%02d", " ", month, day);
106. }
107. } **else** {
108. **if** (week == 6 || week == 0) {
109. printf("%9d!", day);
110. } **else** {
111. printf("%10d", day);
112. }
113. }
114. }
116. **void** printoneWeek(**int** year, **int** weekSeqOfYear, **int** weekSeqShow)
117. {
118. printf("[%02d]", weekSeqShow);
120. **int** sStartSeqOfYear = 7 \* (weekSeqOfYear - 1) - getDaySeqOnJan1(year) + 2;
121. **int** sEndSeqOfYear = sStartSeqOfYear + 6;
122. **int** currentyear = 365 + isLeapYear(year);
123. **int** daySeqOfYear = sStartSeqOfYear;
125. **if** (weekSeqShow == 1) {
126. printOneDay(year, daySeqOfYear, DATE\_INFO\_FULL);
127. daySeqOfYear++;
128. }
130. **for** (; daySeqOfYear <= sEndSeqOfYear; daySeqOfYear++) {
131. **if** (daySeqOfYear <= currentyear) {
132. printOneDay(year, daySeqOfYear, DATE\_INFO\_BRIEF);
133. } **else** {
134. printOneDay(year + 1, daySeqOfYear - currentyear, DATE\_INFO\_BRIEF);
135. }
136. }
137. printf("\n");
138. }
139. date.h
140. #ifndef DATE\_H\_INCLUDED
141. #define DATE\_H\_INCLUDED
143. #define YEAR\_MIN 2019
144. #define YEAR\_MAX 2026
145. #define YEAR\_NUM 8
146. #define DATE\_INFO\_BRIEF 0
147. #define DATE\_INFO\_FULL 1
149. **int** isLeapYear( **int** year );
150. **int** getDaySeqOnJan1( **int** year );
152. /\*int getMonthLength( int year , int month );\*/
153. **int** getDaySeq( **int** year, **int** month, **int** day );
154. **int** getWeekSeqOfYear( **int** year, **int** month, **int** day );
156. **int** getMonth( **int** year, **int** daySeqOfYear );
157. **int** getDay( **int** year, **int** daySeqOfYear );
158. **int** getDaySeqOfWeek( **int** year,**int** daySeqOfYear );
160. **int** getNextMonday( **int** year, **int** daySeqOfYear );
161. **int** getThisMonday ( **int** year, **int** day );
162. **int** getThisSunday( **int** year, **int** daySeqOfYear );
164. **void** setYearArray( **int** Years[], **int** yearnum, **int** year );
166. #endif // DATE\_H\_INCLUDED
167. date.c
168. #include <stdio.h>
169. #include <stdlib.h>
170. #include "date.h"
171. #include "funs.h"
173. #define MONTH\_NUM 12
175. **int** getDay(**int** year, **int** daySeqOfYear)
176. {
177. **extern** **int**  Month\_LEAP\_YEAR[12];
178. **extern** **int**  Month\_NORMAL\_YEAR[12];
179. **int** i = 0;
180. **int** ret = daySeqOfYear;
181. **if** (isLeapYear(year) == 1) {
182. **while** (ret > Month\_LEAP\_YEAR[i]) {
183. ret -= Month\_LEAP\_YEAR[i];
184. i++;
185. }
186. } **else** {
187. **while** (ret > Month\_NORMAL\_YEAR[i]) {
188. ret -= Month\_NORMAL\_YEAR[i];
189. i++;
190. }
191. }
192. **return** ret;
193. }
195. **int** getDaySeq(**int** year, **int** month, **int** day)
196. {
197. **switch** (month) {
198. **case** 12:
199. day += 30;
200. **case** 11:
201. day += 31;
202. **case** 10:
203. day += 30;
204. **case** 9:
205. day += 31;
206. **case** 8:
207. day += 31;
208. **case** 7:
209. day += 30;
210. **case** 6:
211. day += 31;
212. **case** 5:
213. day += 30;
214. **case** 4:
215. day += 31;
216. **case** 3:
217. day += 28 + isLeapYear(year);
218. **case** 2:
219. day += 31;
220. **break**;
221. }
222. **return** day;
223. }
225. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
226. {
227. daySeqOfYear += getDaySeqOnJan1(year) - 1;
228. daySeqOfYear = daySeqOfYear % 7;
229. **return** daySeqOfYear;
230. }
232. **int** getDaySeqOnJan1(**int** year)
233. {
234. **int** result;
235. result = (year - 1 + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400) % 7 + 1;
236. **return** result;
237. }
239. **int** getMonth(**int** year, **int** dayseq)
240. {
241. **extern** **int**  Month\_LEAP\_YEAR[12];
242. **extern** **int**  Month\_NORMAL\_YEAR[12];
243. **int** temp = 1;
244. **if** (isLeapYear(year) == 1) {
245. **while** (dayseq > Month\_LEAP\_YEAR[temp - 1]) {
246. dayseq -= Month\_LEAP\_YEAR[temp - 1];
247. temp++;
248. }
249. } **else** {
250. **while** (dayseq > Month\_NORMAL\_YEAR[temp - 1]) {
251. dayseq -= Month\_NORMAL\_YEAR[temp - 1];
252. temp++;
253. }
254. }
255. **return** temp;
256. }
258. **int** getNextMonday(**int** year, **int** day)
259. {
260. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 1) {
261. **if** ((day + getDaySeqOnJan1(year) - 1) % 7 == 0) {
262. day++;
263. } **else** {
264. day += (7 - (day + getDaySeqOnJan1(year) - 1) % 7) + 1;
265. }
266. }
267. **return** day;
268. }
270. **int** getThisMonday(**int** year, **int** day)
271. {
272. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 1) {
273. **if** ((day + getDaySeqOnJan1(year) - 1) % 7 == 0) {
274. day -= 6;
275. } **else** {
276. day -= ((day + getDaySeqOnJan1(year) - 1) % 7 - 1);
277. }
278. }
279. **return** day;
280. }
282. **int** getThisSunday(**int** year, **int** day)
283. {
284. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 0) {
285. day += 7 - ((day + getDaySeqOnJan1(year) - 1) % 7);
286. }
287. **return** day;
288. }
290. **int** isLeapYear(**int** year)
291. {
292. **if** ((year % 4 == 0 && year % 100!= 0) || year % 400 == 0) {
293. **return** 1;
294. }
295. **return** 0;
296. }
298. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day)
299. {
300. **int** week;
301. **int** daySeqOfYear = getDaySeq(year, month, day);
302. **int** currentyear = 365 + isLeapYear(year);
303. **if** (daySeqOfYear > currentyear) {
304. **if** (getDaySeqOfWeek(year + 1, daySeqOfYear - currentyear) == 0) {
305. week = (daySeqOfYear + getDaySeqOnJan1(year)) / 7;
306. } **else** {
307. week = (daySeqOfYear + getDaySeqOnJan1(year) + (7 - getDaySeqOfWeek(year + 1, daySeqOfYear - currentyear))) / 7;
308. }
309. } **else** {
310. **if** (getDaySeqOfWeek(year, daySeqOfYear) == 0) {
311. week = (daySeqOfYear + getDaySeqOnJan1(year)) / 7;
312. } **else** {
313. week = (daySeqOfYear + getDaySeqOnJan1(year) + (7 - getDaySeqOfWeek(year, daySeqOfYear))) / 7;
314. }
315. }
316. **return** week;
317. }
319. **void** setYearArray(**int** Years[], **int** yearnum, **int** year)
320. {
321. **int** num1;
322. **for** (num1 = 0; num1 < yearnum; num1++, year++) {
323. Years[num1] = year;
324. }
325. }

# 实验三（Calendar09. 寻找生日（结构体））

1. main.c
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include "date.h"
5. #include "funs.h"
7. **const** **int** Years[5] = {2024, 2025, 2026, 2027, 2028};
8. **const** **int** Month\_NORMAL\_YEAR[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
9. **const** **int** Month\_LEAP\_YEAR[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
11. **int** main() {
12. **int** inputMonth, inputDay;
13. **int** i = 0;
14. Day birthDay = {0};
15. Day prepareDay = {0};
16. Day printDay = {0};
17. Day prepareday = {0};
19. printf("please input month: \n");
20. scanf("%d", &inputMonth);
21. printf("please input day: \n");
22. scanf("%d", &inputDay);
24. **for** (i = 0; i < YEAR\_NUM; i++) {
25. birthDay = setDay(Years[i], inputMonth, inputDay);
26. prepareDay = getDayBefore(birthDay, printDayRange);
27. prepareday = prepareDay;
28. printf("%s%s%d\n", " ", "Birthday in Year ", Years[i]);
29. **if** (isDay(Years[i], inputMonth, inputDay)!= 1) {
30. printf(" Not found.\n");
31. **continue**;
32. }
33. printDay = setDay(Years[i], inputMonth, inputDay + i);
34. printoneWeek(prepareday);
35. printf("\n\n");
36. }
37. **return** 0;
38. }
39. funs.h
40. #ifndef FUNS\_H\_INCLUDED
41. #define FUNS\_H\_INCLUDED
43. //void printOneDay( int year, int daySeqOfYear , int formatType );
44. //void printoneWeek(int year, int weekSeqOfYear, int weekSeqShow );
45. //void printOneDay( Day day);
46. **void** printoneWeek( Day day);
48. #endif // FUNS\_H\_INCLUDED
49. funs.c
50. #include <stdio.h>
51. #include <stdlib.h>
52. #include "date.h"
53. #include "funs.h"
55. **extern** **int** Month\_NORMAL\_YEAR[12];
56. **extern** **int** Month\_LEAP\_YEAR[12];
58. **void** printoneWeek(Day prepareday) {
59. Day remain = prepareday;
60. **int** i = 0;
61. **int** index = 1;
62. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
63. **if** (remain.dayseq + 7 - remain.weekDay > isLeapYear(prepareday.year) + 365) {
64. prepareday.weekseq = 1;
65. }
66. printf("[%02d]", prepareday.weekseq);
67. **if** (i == 0) {
68. **int** j;
69. **for** (j = 1; j < remain.weekDay; j++) {
70. printf("%10s", " ");
71. }
72. }
73. **int** k;
74. **for** (k = 0; k <= printDayRange; k++) {
75. **if** (k == 0) {
76. printf("  %02d.%02d.%02d", prepareday.year - 100 \* (prepareday.year / 100), prepareday.month, prepareday.day);
77. prepareday.day++;
78. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
79. **continue**;
80. }
81. **if** (k + remain.weekDay > 7 \* index) {
82. getWeekSeqOfYear(prepareday.year, prepareday.month, getThisSunday(prepareday.year, prepareday.dayseq));
83. **if** (prepareday.dayseq + 7 > isLeapYear(prepareday.year) + 365) {
84. prepareday.weekseq = 1;
85. }
86. printf("\n");
87. printf("[%02d]", prepareday.weekseq);
88. index++;
89. }
90. **if** (k == printDayRange) {
91. **if** (remain.dayseq + k > isLeapYear(prepareday.year) + 365) {
92. prepareday.month = 1;
93. prepareday.day = 1;
94. prepareday.year++;
95. prepareday.dayseq = 1;
96. prepareday.weekseq = 0;
97. printf(" %02d.%02d.%02d\*", prepareday.year - 100 \* (prepareday.year / 100), prepareday.month, prepareday.day);
98. prepareday.day++;
99. prepareday.dayseq++;
100. remain.dayseq = 2;
101. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
102. **continue**;
103. } **else** **if** (prepareday.month == 2 && isLeapYear(prepareday.year) == 1 && prepareday.day == 29) {
104. printf("     %02d.%02d\*", prepareday.month, prepareday.day);
105. prepareday.month++;
106. prepareday.day = 1;
107. prepareday.dayseq++;
108. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
109. **continue**;
110. } **else** **if** (prepareday.day > Month\_NORMAL\_YEAR[prepareday.month - 1]) {
111. prepareday.day = 1;
112. prepareday.month++;
113. printf("    %02d.%02d\*", prepareday.month, prepareday.day);
114. prepareday.day++;
115. prepareday.dayseq++;
116. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
117. **continue**;
118. } **else** {
119. printf("     %02d,%02d\*", prepareday.month, prepareday.day);
120. prepareday.day++;
121. prepareday.dayseq++;
122. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
123. **continue**;
124. }
125. }
126. **if** (remain.dayseq + k > isLeapYear(prepareday.year) + 365) {
127. prepareday.month = 1;
128. prepareday.day = 1;
129. prepareday.year++;
130. prepareday.dayseq = 1;
131. prepareday.weekseq = 0;
132. printf("  %02d.%02d.%02d", prepareday.year - 100 \* (prepareday.year / 100), prepareday.month, prepareday.day);
133. prepareday.day++;
134. prepareday.dayseq++;
135. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
136. remain.dayseq = 2;
137. **continue**;
138. } **else** **if** (prepareday.month == 2 && isLeapYear(prepareday.year) == 1 && prepareday.day == 29) {
139. printf("        %02d", prepareday.day);
140. prepareday.month++;
141. prepareday.day = 1;
142. prepareday.dayseq++;
143. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
144. **continue**;
145. } **else** **if** (prepareday.day > Month\_NORMAL\_YEAR[prepareday.month - 1]) {
146. prepareday.day = 1;
147. prepareday.month++;
148. printf("     %02d.%02d", prepareday.month, prepareday.day);
149. prepareday.day++;
150. prepareday.dayseq++;
151. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
152. **continue**;
153. } **else** {
154. printf("        %02d", prepareday.day);
155. prepareday.dayseq += 1;
156. prepareday.day++;
157. prepareday.weekseq = getWeekSeqOfYear(prepareday.year, prepareday.month, prepareday.day);
158. **if** (prepareday.dayseq > isLeapYear(remain.year) + 365) {
159. prepareday.weekseq = 1;
160. }
161. **continue**;
162. }
163. }
164. }
165. date.h
166. #ifndef DATE\_H\_INCLUDED
167. #define DATE\_H\_INCLUDED
169. #define YEAR\_MIN 2000
170. #define YEAR\_MAX 2030
171. #define YEAR\_NUM 5
172. #define DATE\_INFO\_BRIEF 0
173. #define DATE\_INFO\_FULL 1
174. #define printDayRange 3
176. // 定义 Day 结构体，用于存储日期相关信息
177. **typedef** **struct** day {
178. **int** year;
179. **int** dayseq;
180. **int** month;
181. **int** day;
182. **int** weekseq;
183. **int** weekDay;
184. } Day;
186. // 函数声明
187. **int** isLeapYear(**int** year);
188. **int** getDaySeqOnJan1(**int** year);
189. **int** getMonthLength(**int** year, **int** month);
190. **int** getDaySeq(**int** year, **int** month, **int** day);
191. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day);
192. **int** getMonth(**int** year, **int** daySeqOfYear);
193. **int** getDay(**int** year, **int** daySeqOfYear);
194. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear);
195. **int** getNextMonday(**int** year, **int** daySeqOfYear);
196. **int** getThisMonday(**int** year, **int** day);
197. **int** getThisSunday(**int** year, **int** daySeqOfYear);
198. **void** setYearArray(**int** Years[], **int** yearnum, **int** year);
199. **int** getweekDay(**int** year, **int** daySeq);
200. **int** isDay(**int** year, **int** month, **int** day);
201. Day setDay(**int** year, **int** month, **int** day);
202. Day getDayBefore(Day currentDay, **int** interval);
203. Day getDayAfter(Day currentDay, **int** interval);
204. **int** getTwoDaysInterval(Day startDay, Day endDay);
205. **void** printDay(Day currentDay, **int** displayFormat);
207. #endif // DATE\_H\_INCLUDED
208. date.c
209. #include <stdio.h>
210. #include <stdlib.h>
211. #include "date.h"
212. #include "funs.h"
214. #define MONTH\_NUM 12
216. **int** getDay(**int** year, **int** daySeqOfYear)
217. {
218. **extern** **int**  Month\_LEAP\_YEAR[12];
219. **extern** **int**  Month\_NORMAL\_YEAR[12];
220. **int** i = 0;
221. **int** ret = daySeqOfYear;
222. **if** (isLeapYear(year) == 1) {
223. **while** (ret > Month\_LEAP\_YEAR[i]) {
224. ret -= Month\_LEAP\_YEAR[i];
225. i++;
226. }
227. } **else** {
228. **while** (ret > Month\_NORMAL\_YEAR[i]) {
229. ret -= Month\_NORMAL\_YEAR[i];
230. i++;
231. }
232. }
233. **return** ret;
234. }
236. **int** getDaySeq(**int** year, **int** month, **int** day)
237. {
238. **switch** (month) {
239. **case** 12:
240. day += 30;
241. **case** 11:
242. day += 31;
243. **case** 10:
244. day += 30;
245. **case** 9:
246. day += 31;
247. **case** 8:
248. day += 31;
249. **case** 7:
250. day += 30;
251. **case** 6:
252. day += 31;
253. **case** 5:
254. day += 30;
255. **case** 4:
256. day += 31;
257. **case** 3:
258. day += 28 + isLeapYear(year);
259. **case** 2:
260. day += 31;
261. **break**;
262. }
263. **return** day;
264. }
266. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
267. {
268. daySeqOfYear += getDaySeqOnJan1(year) - 1;
269. daySeqOfYear = daySeqOfYear % 7;
270. **return** daySeqOfYear;
271. }
273. **int** getDaySeqOnJan1(**int** year)
274. {
275. **int** result;
276. result = (year - 1 + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400) % 7 + 1;
277. **return** result;
278. }
280. **int** getMonth(**int** year, **int** dayseq)
281. {
282. **extern** **int**  Month\_LEAP\_YEAR[12];
283. **extern** **int**  Month\_NORMAL\_YEAR[12];
284. **int** temp = 1;
285. **if** (isLeapYear(year) == 1) {
286. **while** (dayseq > Month\_LEAP\_YEAR[temp - 1]) {
287. dayseq -= Month\_LEAP\_YEAR[temp - 1];
288. temp++;
289. }
290. } **else** {
291. **while** (dayseq > Month\_NORMAL\_YEAR[temp - 1]) {
292. dayseq -= Month\_NORMAL\_YEAR[temp - 1];
293. temp++;
294. }
295. }
296. **return** temp;
297. }
299. **int** getNextMonday(**int** year, **int** day)
300. {
301. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 1) {
302. **if** ((day + getDaySeqOnJan1(year) - 1) % 7 == 0) {
303. day++;
304. } **else** {
305. day += (7 - (day + getDaySeqOnJan1(year) - 1) % 7) + 1;
306. }
307. }
308. **return** day;
309. }
311. **int** getThisMonday(**int** year, **int** day)
312. {
313. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 1) {
314. **if** ((day + getDaySeqOnJan1(year) - 1) % 7 == 0) {
315. day -= 6;
316. } **else** {
317. day -= ((day + getDaySeqOnJan1(year) - 1) % 7 - 1);
318. }
319. }
320. **return** day;
321. }
323. **int** getThisSunday(**int** year, **int** day)
324. {
325. **if** ((day + getDaySeqOnJan1(year) - 1) % 7!= 0) {
326. day += 7 - ((day + getDaySeqOnJan1(year) - 1) % 7);
327. }
328. **return** day;
329. }
331. **int** isLeapYear(**int** year)
332. {
333. **if** ((year % 4 == 0 && year % 100!= 0) || year % 400 == 0) {
334. **return** 1;
335. }
336. **return** 0;
337. }
339. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day)
340. {
341. **int** week;
342. **int** daySeqOfYear = getDaySeq(year, month, day);
343. **int** currentyear = 365 + isLeapYear(year);
344. **if** (daySeqOfYear > currentyear) {
345. **if** (getDaySeqOfWeek(year + 1, daySeqOfYear - currentyear) == 0) {
346. week = (daySeqOfYear + getDaySeqOnJan1(year)) / 7;
347. } **else** {
348. week = (daySeqOfYear + getDaySeqOnJan1(year) + (7 - getDaySeqOfWeek(year + 1, daySeqOfYear - currentyear))) / 7;
349. }
350. } **else** {
351. **if** (getDaySeqOfWeek(year, daySeqOfYear) == 0) {
352. week = (daySeqOfYear + getDaySeqOnJan1(year)) / 7;
353. } **else** {
354. week = (daySeqOfYear + getDaySeqOnJan1(year) + (7 - getDaySeqOfWeek(year, daySeqOfYear))) / 7;
355. }
356. }
357. **return** week;
358. }
360. **void** setYearArray(**int** Years[], **int** yearnum, **int** year)
361. {
362. **int** num1;
363. **for** (num1 = 0; num1 < yearnum; num1++, year++) {
364. Years[num1] = year;
365. }
366. }