

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 | 崔曼妮 | 学号 | U202413702 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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：visual studio

(因为日历系列有的实验比较相似，前面的实验比较简单，因此挑了四个典型的写，并且运用动态分配内存，以及自学的链表部分改进了实验三，具体步骤在实验四）

# 实验一

## 实验任务

Calendar07.打印全学年校历(函数版)

·[实验目标]

-巩固函数的使用,对现有函数进行功能升级(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]

-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

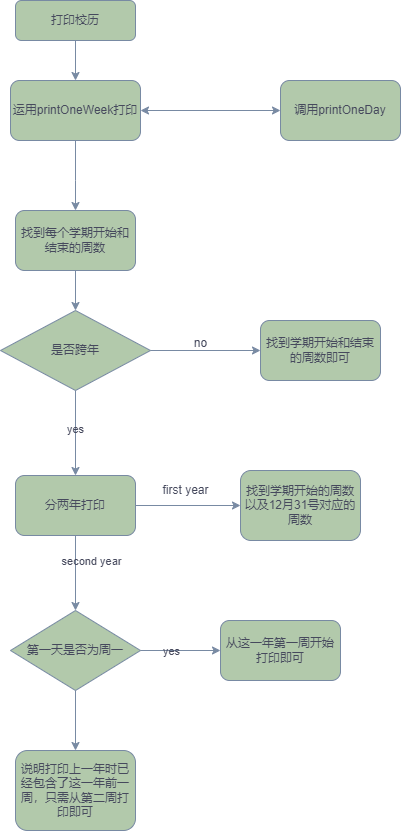
-约定秋季学期从当年9月4日所在周的周一开始,到第二年1月20日之前一

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。

-要求设计一个打印某周周历的函数,支持不同学期校历的打印需求;改进

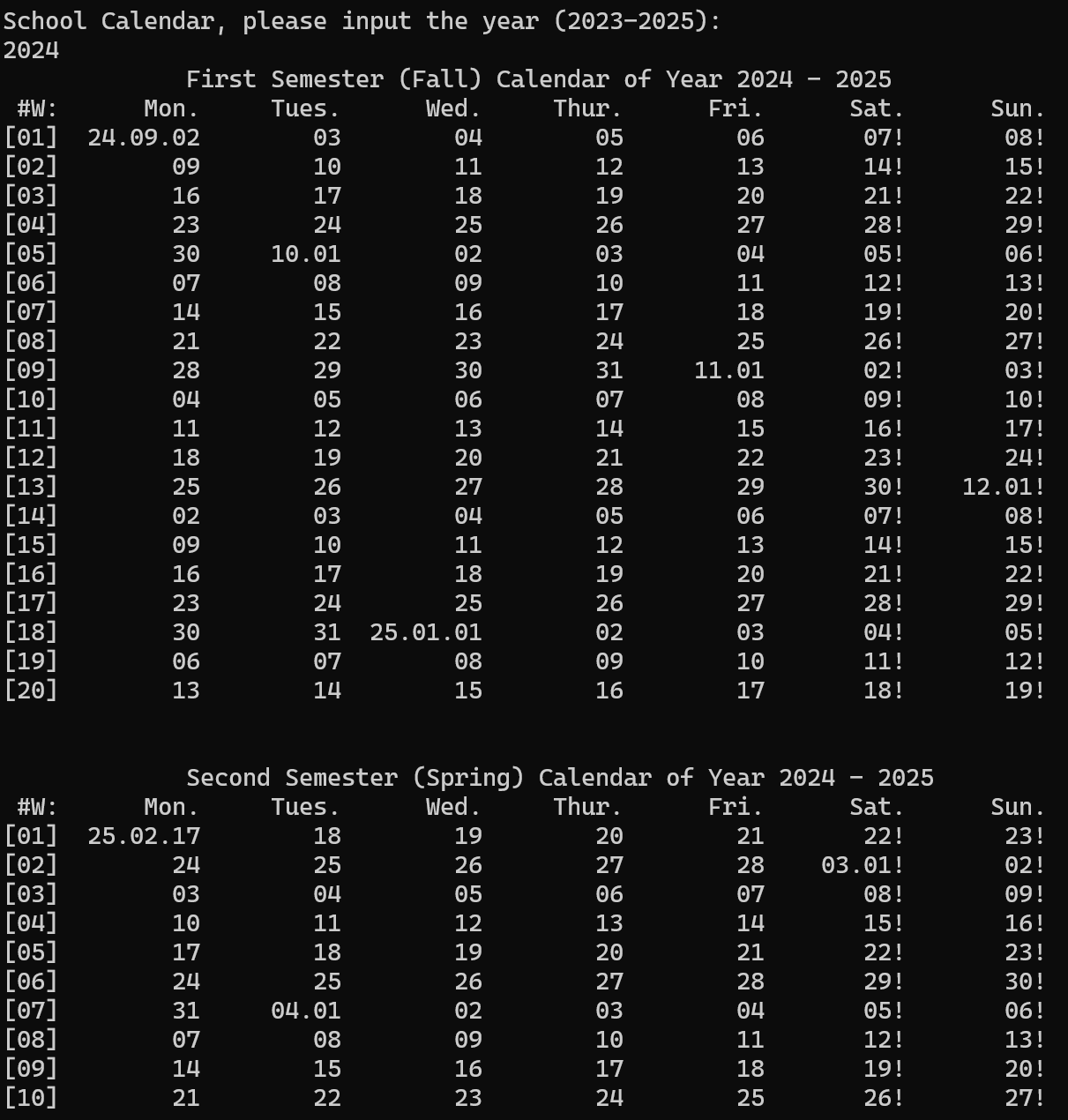
打印某日的函数,支持在校历首日和跨年日显示年月日信息。

## 实验步骤

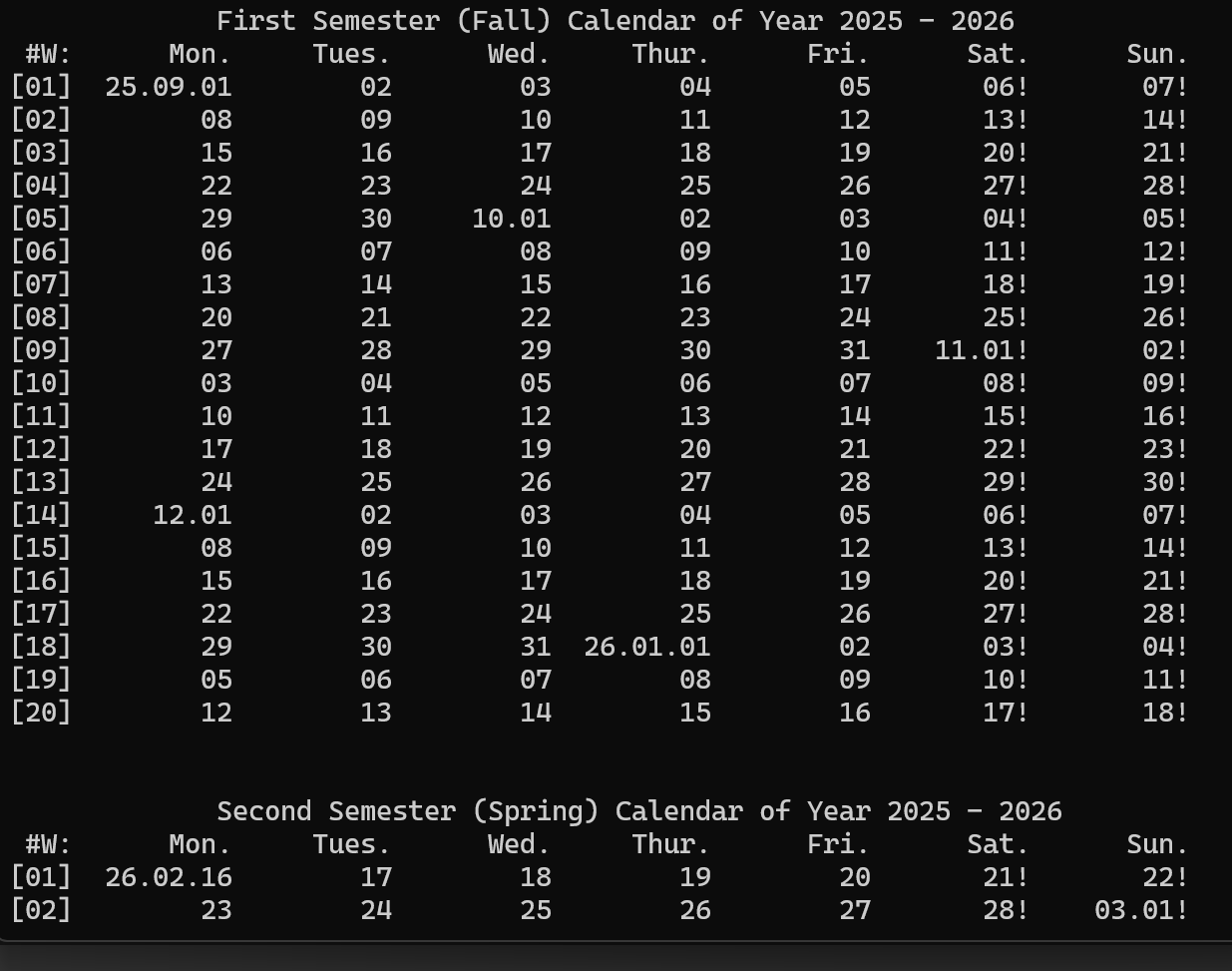


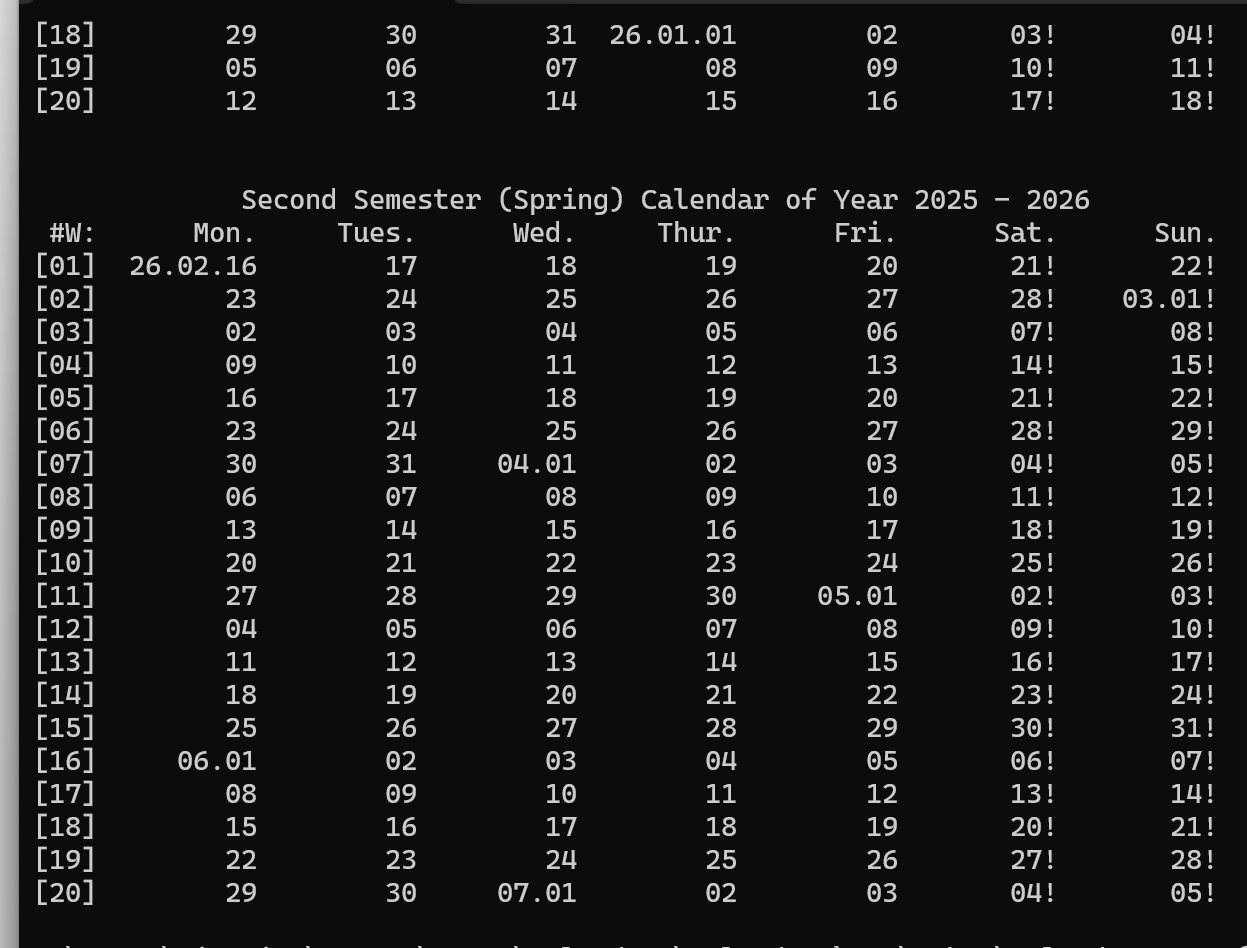
## 代码测试

输入2024



输入2025





## 实验结论

任务已完成

## 实验总结

独立的C文件和H文件，使查找函数更容易，特定功能的函数封装成一个头文件，利于代码复用和更新

错误：0和O混淆使一直出现未定义该变量的错误

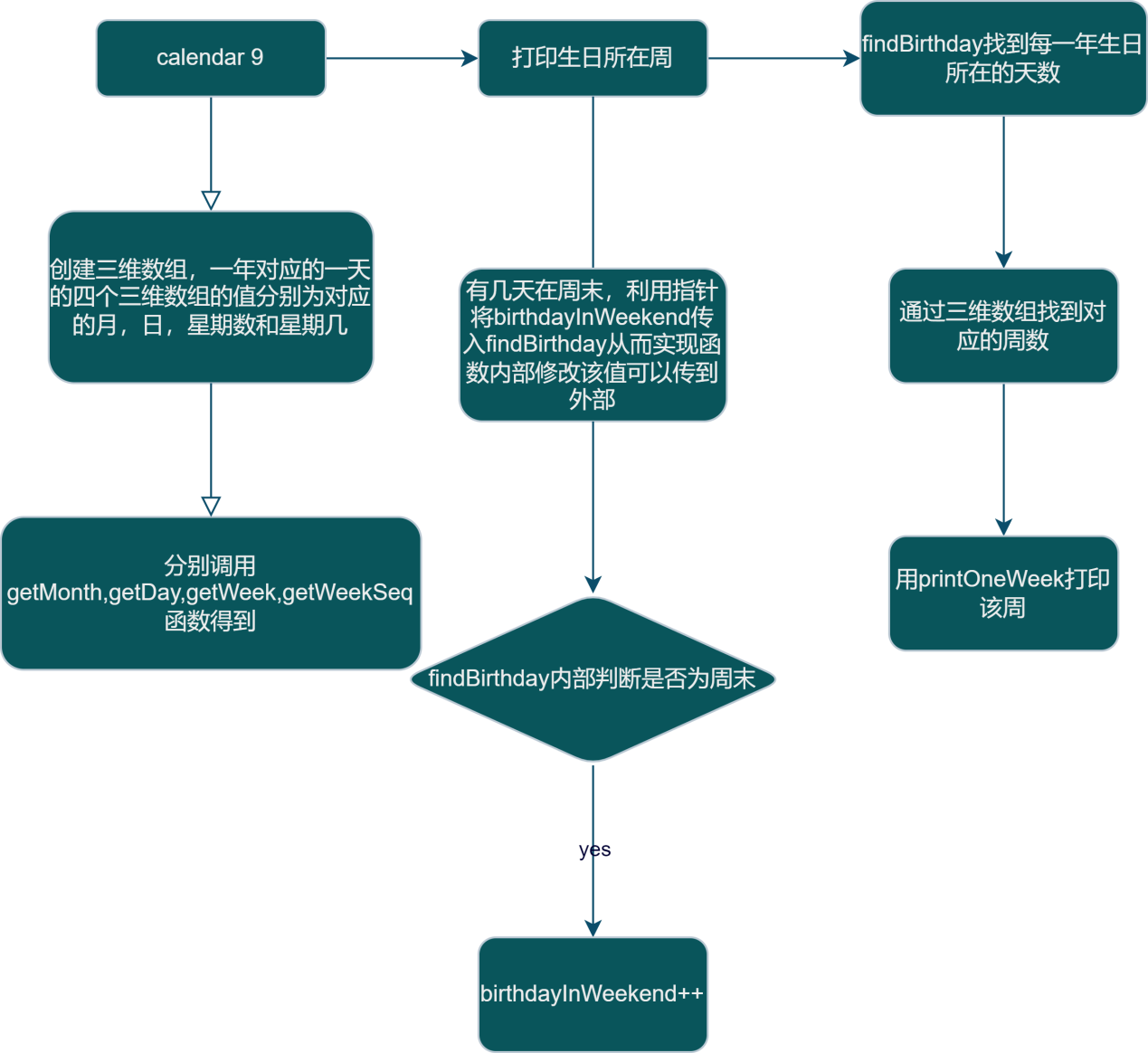
# 实验二

## 实验任务

– 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

## 实验步骤

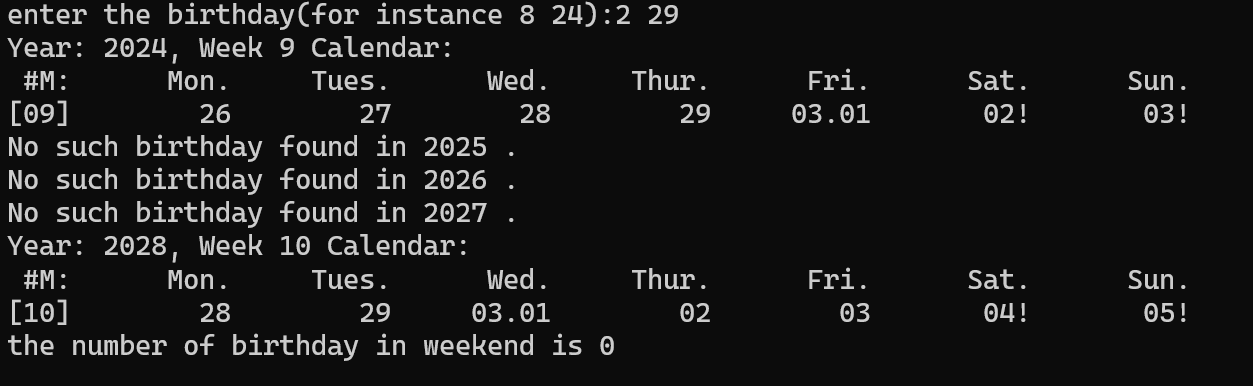


## 代码测试

输入8.24



输入2.29



## 实验结论

完成任务

## 实验总结

利用多维的数组储存对应天数的各个值，使查找和调用都更容易，但是占用的空间会比较大。

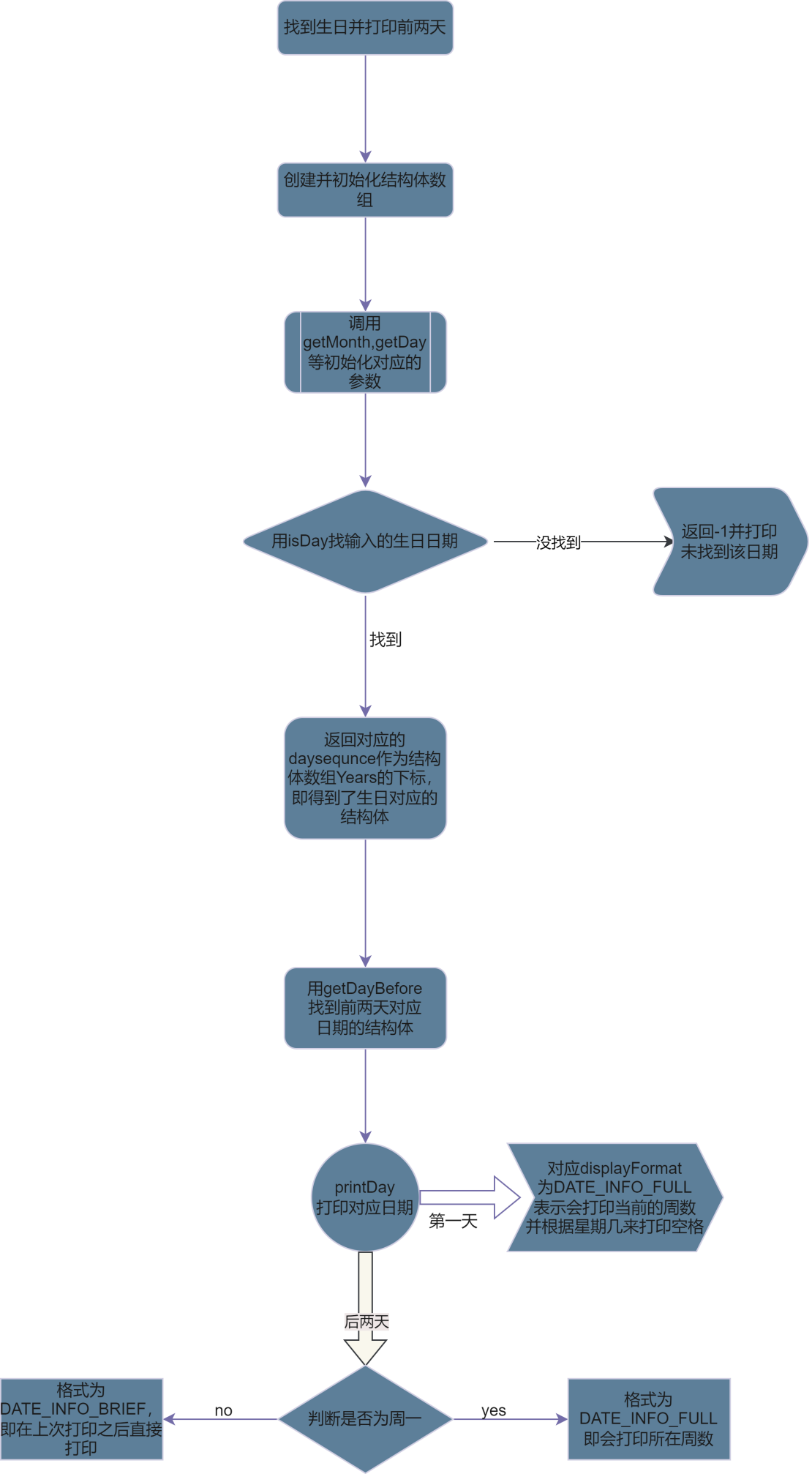
# 实验三

## 实验任务

-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

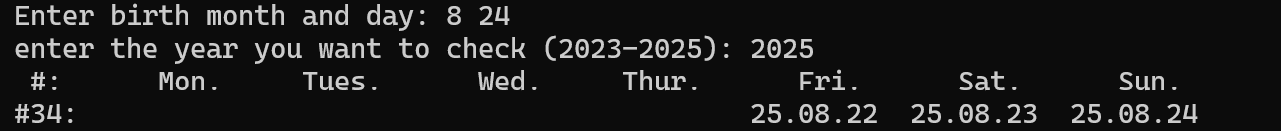
-用日期结构体记录单一日期的所有数据属性

## 实验步骤

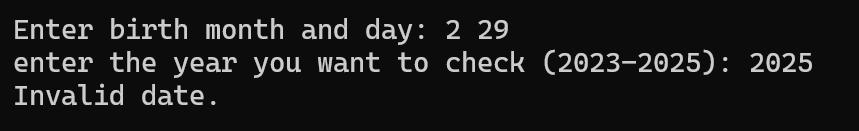


## 代码测试

输入8.24



输入2.29



## 实验结论

任务已完成

## 实验总结

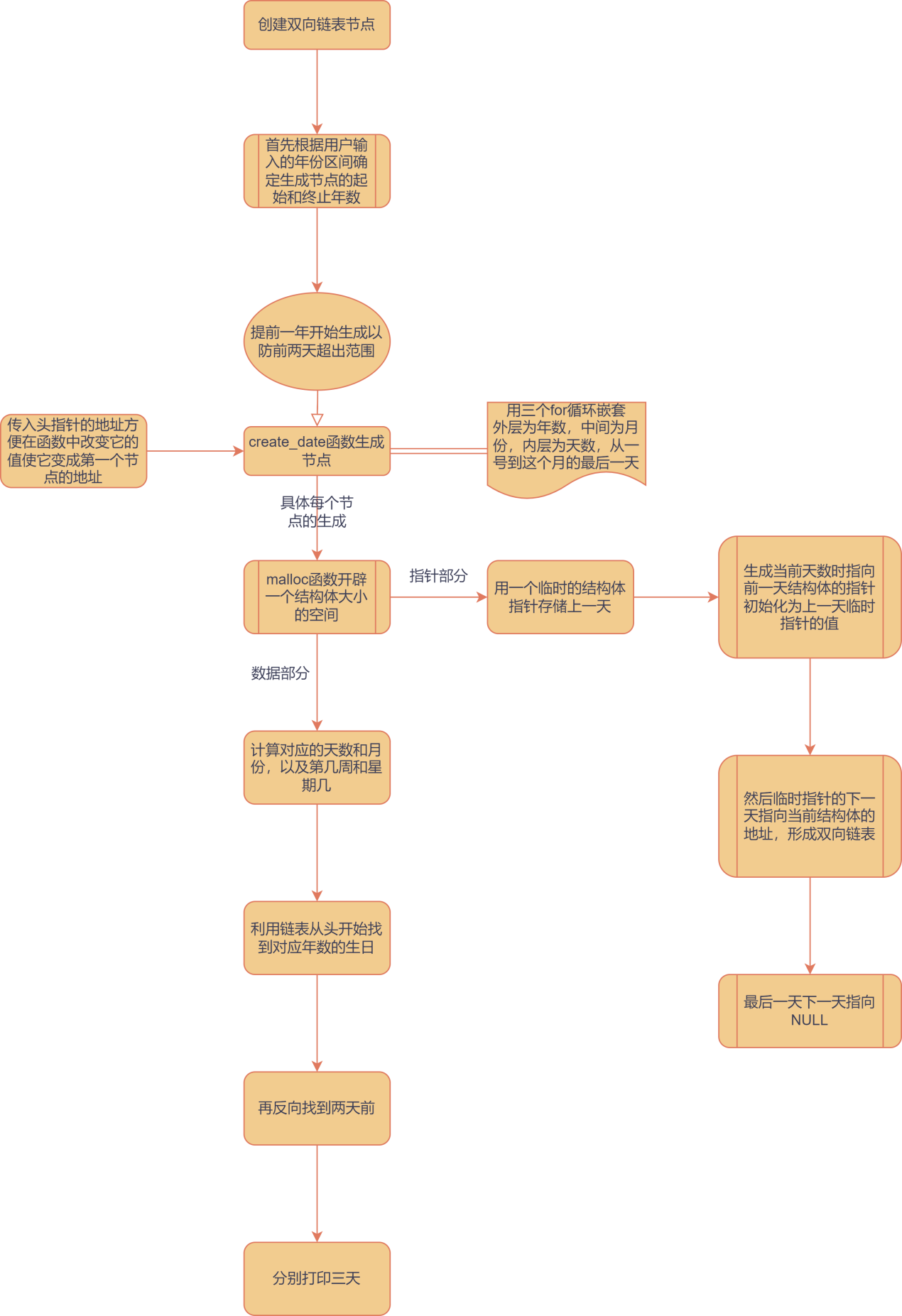
结构体相较于上一次任务中的数组，储存内容的格式更丰富，同时调用的时候根据命名的名称来找，更容易，代码组织更清晰。

# 实验四

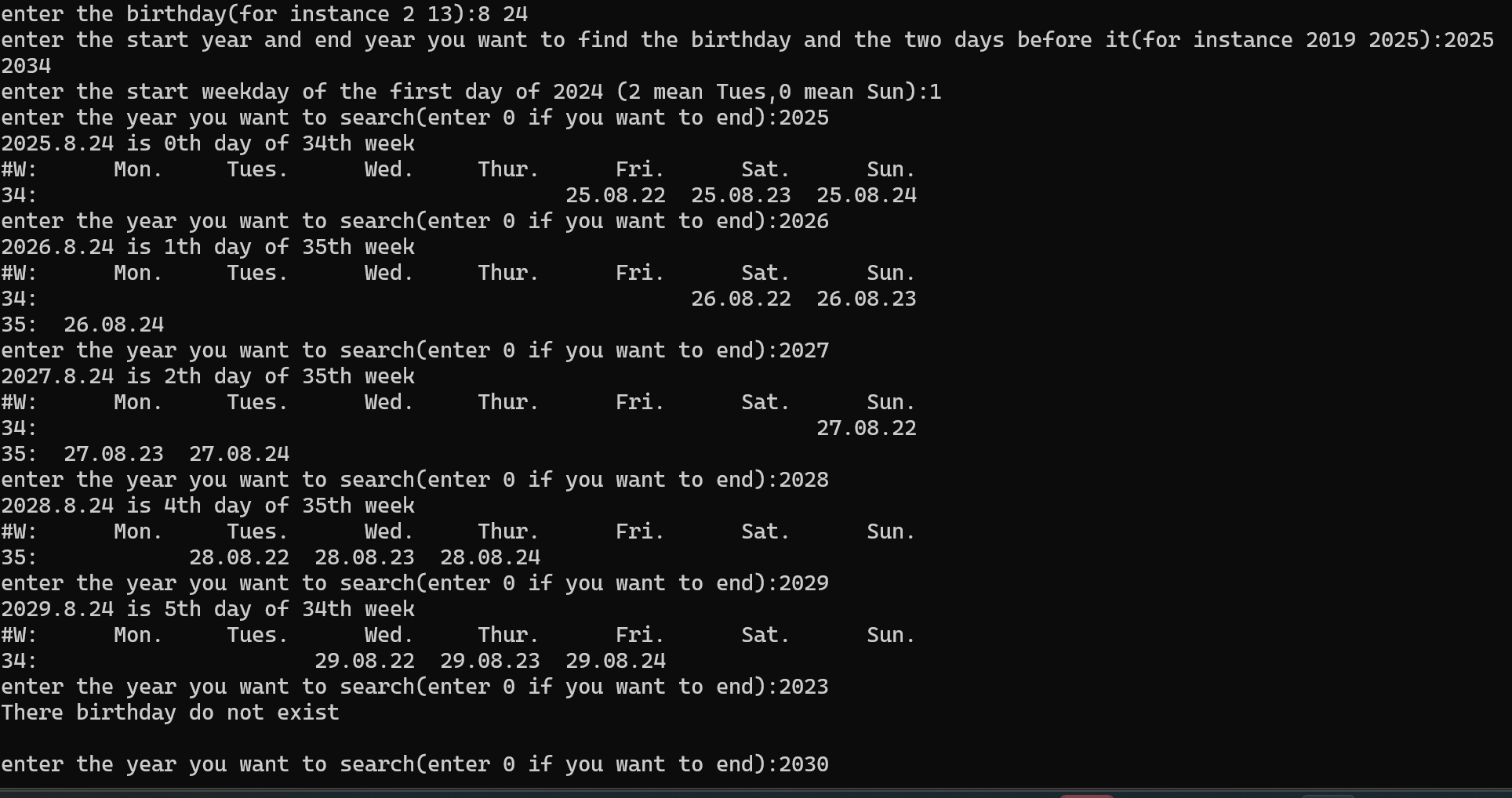
## 实验任务

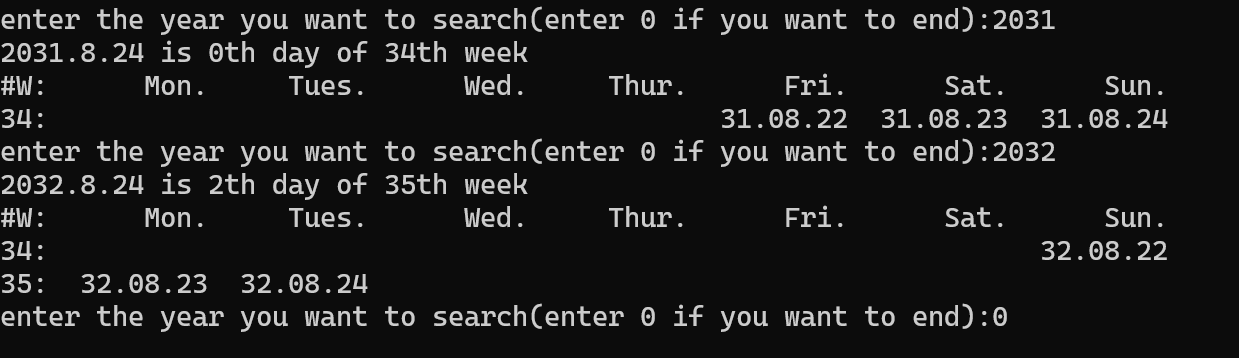
运用动态分配内存和链表的知识，我对实验三代码进行了改进，基本找到生日并打印前两天的任务与实验三相同，可以自己输入想要查询年份的起始和终止年份。然后可以连续查询起始和终止年份区间内年份的生日，直到输入0结束。

## 实验步骤



## 代码测试





## 实验结论

任务已完成

## 实验总结

动态分配内存使链表可以储存任意年数区间的日期

为什么要用双向链表：双向链表使查找对应日期，反向找到两天前，再正向打印三天都更容易，也可以使计算机运行速度更快

错误：把头指针和头指针的地址有些弄混

# 实验五

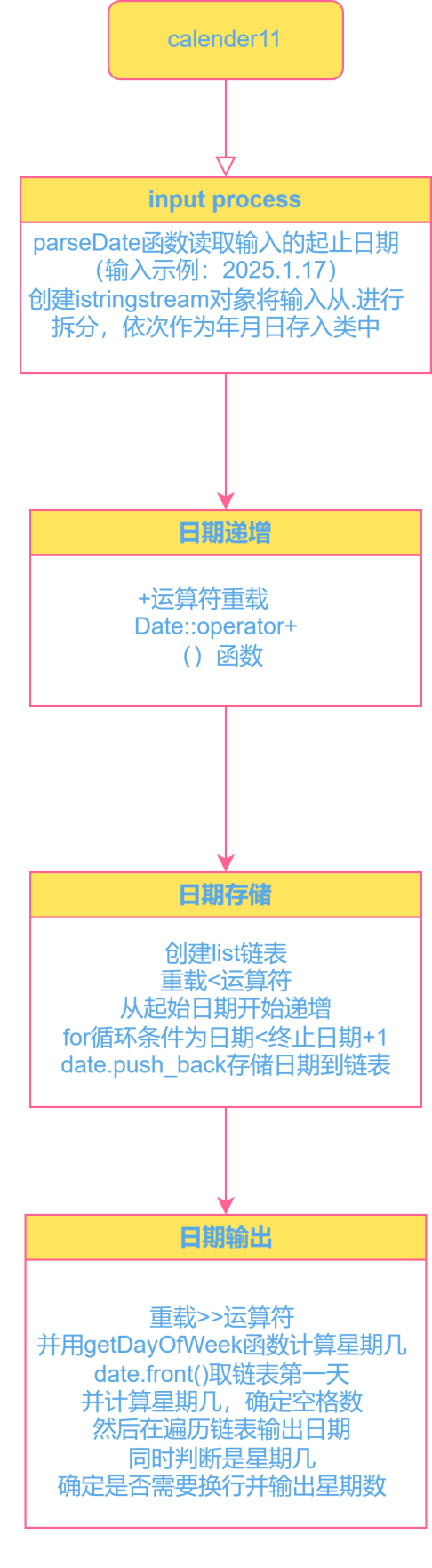
## 实验任务

日历11

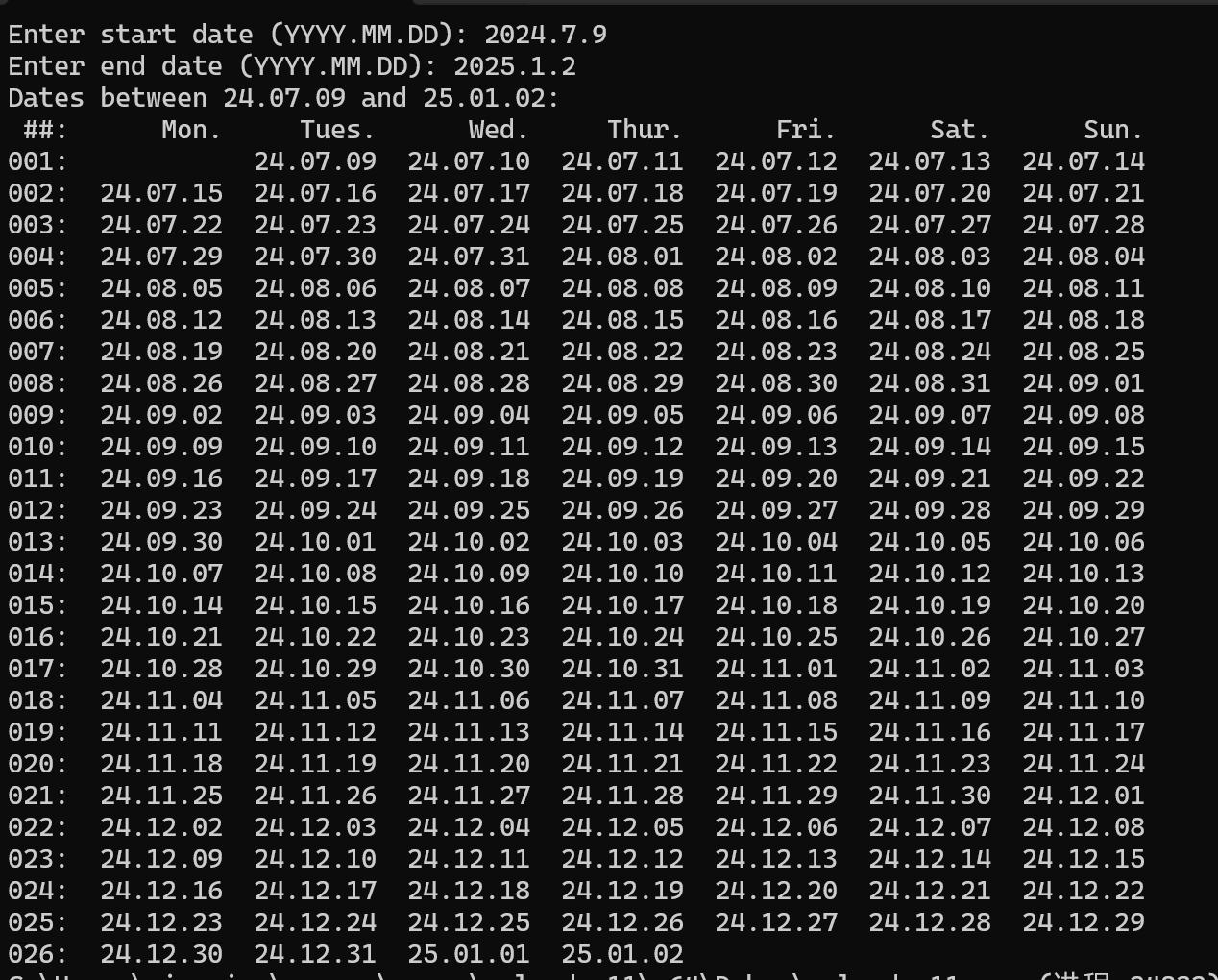
C++结构体加函数，实现面向对象的编程

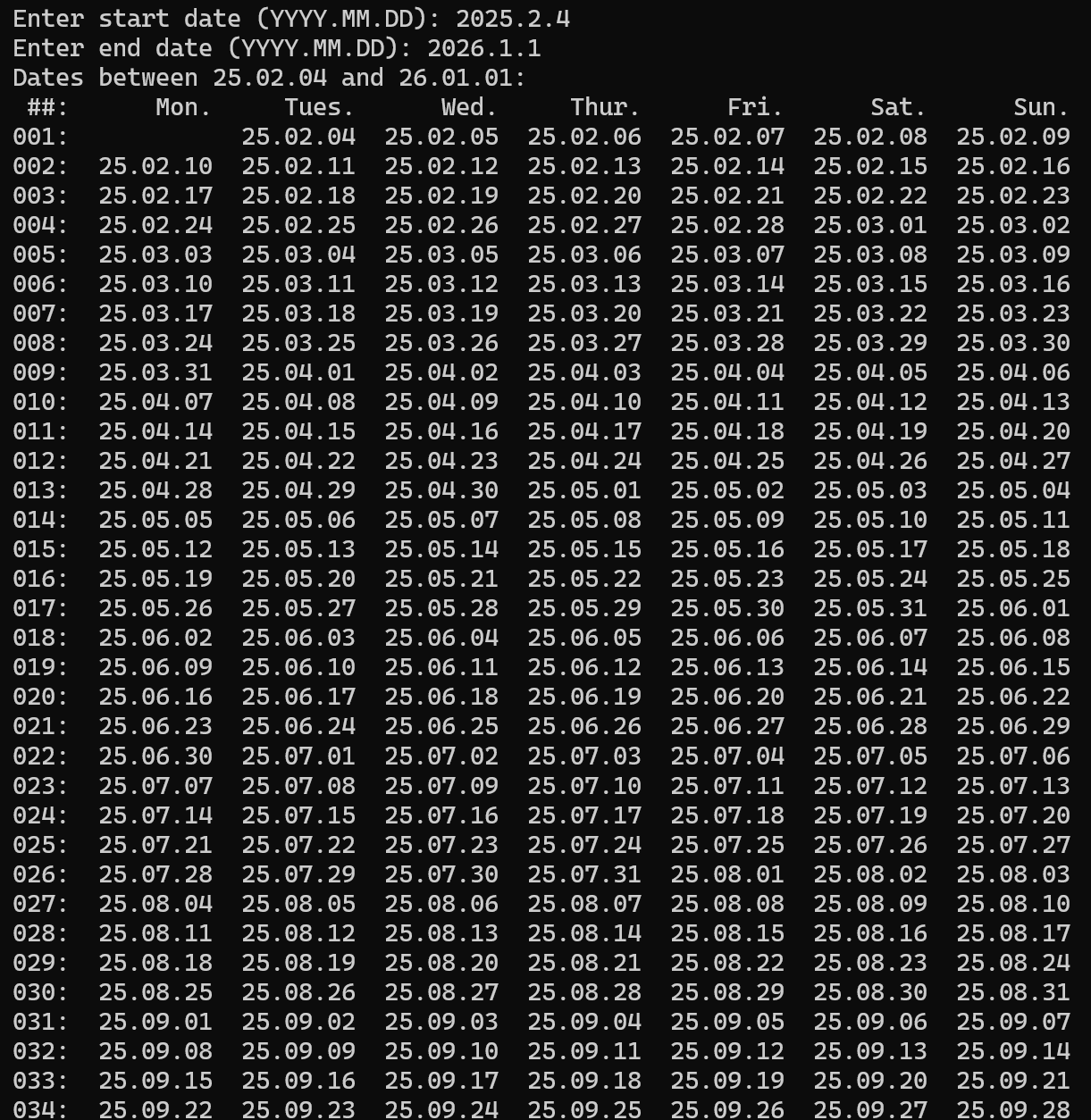
输入起始日期和终止日期，输出中间天数的日历格式

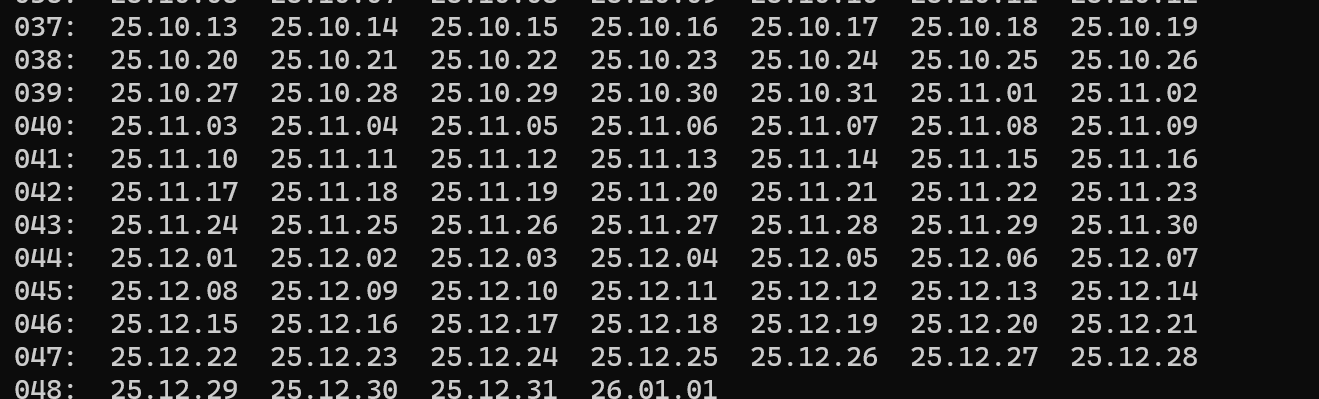
## 实验步骤



## 代码测试







## 实验结论

任务已完成

## 实验总结

利用运算符重载，充分体现面向对象编程的特点，大大提高了代码的可读性。同时利用动态生成的链表来储存结构体，比c语言的数组更方便

# 本课程学习总结

学习C语言的历程和总结（心酸血泪史）：

我从最开始丑陋的格式，没有规矩的命名，喜欢一个main函数写到底，写完再运行，从来不用debug,从来不写注释。到后来由亲身的编代码经历，切身的体会到了这些规矩的重要性。

首先格式虽然麻烦，命名规矩虽然有一些麻烦会多打些字，但是为了代码有更好的可读性，不管是以后自己看还是别人看。并且为了以后的自我发展，以后必然是多人协作开发代码。好的格式不仅让自己看着舒服，也让别人有看下去的欲望。（不过我认为这个不必要自己花时间打空格什么的，可以找个好的编译器或者下个插件）

然后我认为函数化的编程可以很大程度上提高代码的可读性，分成块编也更简单，思考起来更容易，更清晰。此外，如果代码出现了什么错误，打包成函数也可以更容易知道哪个地方错了，改动的也不会特别多，并且debug也会容易很多。

以前我编函数喜欢全部编完再运行，但是这样报错或者出现bug改动会相当困难，并且改动的也会非常多，有的时候甚至和重编没什么区别。所以现在如果我编比较大的程序，我会每个主要的函数编写完成后都会运行测试一下。

其次就是debug功能，以前我代码出错了都是干瞪眼检查，知道我编写那个老师上课提到的打牌程序，瞪眼完全找不到。然后用了一次debug之后我就离不开它了。在一次又一次debug之后，我逐渐掌握了技巧。我一般先在主函数里面主要的函数打断点，运行一下看具体那一步寄了，然后再在有问题的中间打断点或者步入一步一步看，最后找到错误的地方，类似于二分法。对于一些数据不对等情况，我觉得那个watch窗口真的非常好用。除此之外，我也喜欢在函数中printf几个关键的值来确定错误。

还有就是独立的C文件和H文件，本来觉得很没有必要。但在嵌入式方向经常运用这个，我也感受到了这样编会清楚很多，尤其是在有很多函数的情况下，将函数分类存放，便于修改和阅读。

最后就是注释的问题，原来我认为我函数以及变量命名的清楚易懂，无需注释，但是在编写那个打扑克排的程序的时候，每次中间停了下一次编就要把那上千行的代码再看一遍。我也因此感受到了适当注释记录自己思路的重要性。

总之就是在学习c语言的过程中，我越来越感受到老师说的真对。也感受到了从面向过程，到面向数据，到面向对象编程思路的变化。也在一次又一次的痛苦的debug中感受到了作为码农的基本素养（莫急莫慌），磨练了自己的心性。同时c语言的学习基础也让我很快的上手了python语言，debug用的得心应手。同时也让我了解了很多数据结构，更好的上手c++。因此不管我以后会不会经常用到c语言，我都感觉我这段基于自己动手感受的学习经历是很有用的，也感谢老师生动有用的讲解。

# 附录

1. 实验一
2. main.c
3. #include <stdio.h>
4. #include <stdlib.h>
6. #include "date.h"
7. #include "funs.h"

10. **int** main(**void**)
11. {
12. // Declare variables
13. **int** inputYear;
15. // Display the program information
16. printf("School Calendar, please input the year (%d-%d): \n", YEAR\_MIN, YEAR\_MAX);
17. scanf\_s("%d", &inputYear);
19. **if** (inputYear < YEAR\_MIN || inputYear > YEAR\_MAX) {
20. printf("input error!\n");
21. }
23. // Print calendar for fall semester in this year
24. printf("%13s%s%d%s%d\n", " ", "First Semester (Fall) Calendar of Year ", inputYear, " - ", inputYear + 1);
25. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
26. **int** startWeekSeq = getWeekSeq0fYear(inputYear, 9, 4);
27. **int** endWeekSeq = getWeekSeq0fYear(inputYear, 12, 31);
28. **for** (**int** i = startWeekSeq; i <= endWeekSeq; i++) {
29. printOneWeek(inputYear, i, i - startWeekSeq + 1);
30. }
31. inputYear++;
32. startWeekSeq = 1;
33. **if** (getDaySeqOnJan1(inputYear) != 0) {
34. startWeekSeq++;
35. }
36. endWeekSeq = getWeekSeq0fYear(inputYear, 1, 20) - 1;
37. **for** (**int** i = startWeekSeq; i <= endWeekSeq; i++) {
38. printOneWeek(inputYear, i, getWeekSeq0fYear(inputYear - 1, 12, 31) - getWeekSeq0fYear(inputYear - 1, 9, 4) + i - startWeekSeq + 2);
39. }
40. printf("\n\n");
41. // Print calendar for spring semester in next year
42. printf("%13s%s%d%s%d\n", " ", "Second Semester (Spring) Calendar of Year ", inputYear - 1, " - ", inputYear);
43. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
44. startWeekSeq = getWeekSeq0fYear(inputYear, 2, 15) + 1;
45. endWeekSeq = getWeekSeq0fYear(inputYear, 7, 1);
46. **for** (**int** i = startWeekSeq; i <= endWeekSeq; i++) {
47. printOneWeek(inputYear, i, i - startWeekSeq + 1);
48. }
50. **return** 0;
51. }
52. Funs.c
53. #include "funs.h"
54. #include "date.h"
55. #include <stdio.h>
57. **void** printOneDay(**int** year, **int** daySeq0fYear, **int** formatType) {
58. **if** (formatType == DATE\_INFO\_FULL||daySeq0fYear==1) {
59. **if** (getDaySeq0fWeek(year, daySeq0fYear) == 5 || getDaySeq0fWeek(year, daySeq0fYear) == 6) {
60. printf(" %02d.%02d.%02d!", year - 2000, getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
61. }
62. **else** {
63. printf("  %02d.%02d.%02d", year - 2000, getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
64. }
65. }
66. **else** **if** (getDay(year, daySeq0fYear) == 1) {
67. **if** (getDaySeq0fWeek(year, daySeq0fYear) == 5 || getDaySeq0fWeek(year, daySeq0fYear) == 6) {
68. printf("    %02d.%02d!", getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
69. }
70. **else** {
71. printf("     %02d.%02d", getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
72. }
73. }
74. **else** **if** (getDaySeq0fWeek(year, daySeq0fYear) == 5 || getDaySeq0fWeek(year, daySeq0fYear) == 6) {
75. printf("%7s%02d!", " ", getDay(year, daySeq0fYear));
76. }
77. **else** {
78. printf("%8s%02d", " ", getDay(year, daySeq0fYear));
79. }
80. }
82. **void** printOneWeek(**int** year, **int** weekSeq0fYear, **int** weekSeqShow){
84. // Before Monday, print the week sequence
85. printf("[%02d]", weekSeqShow);
87. // Declare variables to navigate the semester
88. **int** sStartSeq0fYear = 7 \* (weekSeq0fYear - 1) - getDaySeqOnJan1(year) + 1;
89. **int** sEndSeq0fYear = sStartSeq0fYear + 6;
90. **int** currentYearLength = (isLeapYear(year) == 1) ? 366 : 365;
91. **int** daySeq0fYear = sStartSeq0fYear;
93. // If this is the first week of semester, show full month and day for first da
94. **if** (weekSeqShow == 1) {
95. printOneDay(year, daySeq0fYear, DATE\_INFO\_FULL);
96. daySeq0fYear++;
97. }

100. // Print the semester calendar
101. **for** (; daySeq0fYear <= sEndSeq0fYear; daySeq0fYear++){
103. // Print the month and day for the current day
104. **if** (daySeq0fYear <= currentYearLength){
105. printOneDay(year, daySeq0fYear, DATE\_INFO\_BRIEF);
107. }
108. **else** {
109. printOneDay(year + 1, daySeq0fYear - currentYearLength, DATE\_INFO\_BRIEF);
110. }
112. }
113. // After Sunday, print a new line
114. printf("\n");
115. }
116. Funs.h
117. #ifndef FUNS\_H
118. #define FUNS\_H
120. // Functions support school calendar display
121. **void** printOneDay(**int** year, **int** daySeq0fYear, **int** formatType);
122. **void** printOneWeek(**int** year, **int** weekSeq0fYear, **int** weekSeqShow);
124. #endif // FUNS\_H\_INCLUDED
125. date.c
126. #include <stdio.h>
127. #include <stdlib.h>
128. #include "date.h"
129. #include "funs.h"
131. **int** isLeapYear(**int** year) {
132. **return** (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
133. }
134. **int** getDaySeqOnJan1(**int** year) {
135. **int** totalDays = 0;
136. **for** (**int** i = 1900; i < year; i++) {
137. totalDays += isLeapYear(i) ? 366 : 365;
138. }
139. **return** totalDays % 7; // 0: 周日, 1: 周一, ..., 6: 周六
140. }
141. **int** getMonthLength(**int** year, **int** month) {
142. **if** (month == 2) {
143. **return** isLeapYear(year) ? 29 : 28;
144. }
145. **else** **if** (month == 4 || month == 6 || month == 9 || month == 11) {
146. **return** 30;
147. }
148. **else** {
149. **return** 31;
150. }
151. }
152. **int** getDaySeq(**int** year, **int** month, **int** day) {
153. **int** totalDays = 0;
154. **for** (**int** i = 1; i < month; i++) {
155. totalDays += getMonthLength(year, i);
156. }
157. totalDays += day;
158. **return** totalDays;
159. }
160. **int** getWeekSeq0fYear(**int** year, **int** month, **int** day) {
161. **return** (getDaySeqOnJan1(year) + getDaySeq(year, month, day) - 1) / 7 + 1;
162. }
163. **int** getMonth(**int** year, **int** daySeq0fYear) {
164. **int** month = 1;
165. **while** (daySeq0fYear > getMonthLength(year, month)) {
166. daySeq0fYear -= getMonthLength(year, month);
167. month++;
168. }
169. **return** month;
170. }
171. **int** getDay(**int** year, **int** daySeq0fYear) {
172. **int** month = getMonth(year, daySeq0fYear);
173. **return** daySeq0fYear - getDaySeq(year, month, 1) + 1;
174. }
175. **int** getDaySeq0fWeek(**int** year, **int** daySeq0fYear) {
176. **return** (getDaySeqOnJan1(year) + daySeq0fYear - 1) % 7;
177. }
178. **int** getNextMonday(**int** year, **int** daySeq0fYear) {
179. **int** daySeq0fWeek = getDaySeq0fWeek(year, daySeq0fYear);
180. **return** daySeq0fYear + (daySeq0fWeek == 0 ? 1 : 8 - daySeq0fWeek);
181. }
182. **int** getThisMonday(**int** year, **int** daySeq0fYear) {
183. **int** daySeq0fWeek = getDaySeq0fWeek(year, daySeq0fYear);
184. **return** daySeq0fYear - (daySeq0fWeek == 0 ? 6 : daySeq0fWeek - 1);
185. }
186. **int** getThisSunday(**int** year, **int** daySeq0fYear) {
187. **int** daySeq0fWeek = getDaySeq0fWeek(year, daySeq0fYear);
188. **return** daySeq0fYear + (daySeq0fWeek == 0 ? 0 : 7 - daySeq0fWeek);
189. }
190. Date.h
191. #ifndef DATE\_H
192. #define DATE\_H
194. #define YEAR\_MIN    2023
195. #define YEAR\_MAX    2025
196. #define DATE\_INFO\_BRIEF  1
197. #define DATE\_INFO\_FULL   0
199. // \*\*\*
200. // Functions used in this program
201. // \*\*\*\*\*\*\*
203. // Functions on different year
204. **int** isLeapYear(**int** year);
205. **int** getDaySeqOnJan1(**int** year);
207. // Functions about month and day
208. **int** getMonthLength(**int** year, **int** month);
209. **int** getDaySeq(**int** year, **int** month, **int** day);
210. **int** getWeekSeq0fYear(**int** year, **int** month, **int** day);
212. // Functions for properties of one day
213. **int** getMonth(**int** year, **int** daySeq0fYear);
214. **int** getDay(**int** year, **int** daySeq0fYear);
215. **int** getDaySeq0fWeek(**int** year, **int** daySeq0fYear);
217. // Functions for day movement calculation
218. **int** getNextMonday(**int** year, **int** daySeq0fYear);
219. **int** getThisMonday(**int** year, **int** daySeq0fYear);
220. **int** getThisSunday(**int** year, **int** daySeq0fYear);
222. #endif
223. 实验二

（1）main.c

1. #include <stdio.h>
3. #define YEAR\_NUM 5
5. #define START\_YEAR 2024
6. #define END\_YEAR 2028
7. // 三维数组，记录多年的日期数据
8. **int** Days[YEAR\_NUM][366][4];
10. // 函数声明
11. **void** initializeDays();
12. **void** findBirthday(**int** year, **int** month, **int** day, **int**\* weekend);
13. **void** printWeekCalendar(**int** year, **int** weekSeq);
14. **int** isLeapYear(**int** year);
15. **int** getDaySeqOnJan1(**int** year);
17. // Functions about month and day
18. **int** getMonthLength(**int** year, **int** month);
19. **int** getDaySeq(**int** year, **int** month, **int** day);
20. **int** getWeekSeq0fYear(**int** year, **int** month, **int** day);
22. // Functions for properties of one day
23. **int** getMonth(**int** year, **int** daySeq0fYear);
24. **int** getDay(**int** year, **int** daySeq0fYear);
25. **int** getDaySeq0fWeek(**int** year, **int** daySeq0fYear);
27. // Functions for day movement calculation
28. **int** getNextMonday(**int** year, **int** daySeq0fYear);
29. **int** getThisMonday(**int** year, **int** daySeq0fYear);
30. **int** getThisSunday(**int** year, **int** daySeq0fYear);
31. **void** printOneWeek(**int** year, **int** weekSeq0fYear);
32. **void** printOneDay(**int** year, **int** daySeq0fYear);
34. **int** main() {
36. initializeDays();
37. **int** month, day;
38. **int** birthdayInWeekend=0;
39. printf("enter the birthday(for instance 8 24):");
40. scanf\_s("%d %d", &month, &day);
41. **for** (**int** i = START\_YEAR; i <= END\_YEAR; i++) {
42. findBirthday(i, month, day,&birthdayInWeekend);
43. }
44. printf("the number of birthday in weekend is %d\n", birthdayInWeekend);
45. **return** 0;
46. }
48. // 初始化日期数组的函数
49. **void** initializeDays() {
50. **for** (**int** i = 0; i < 5; i++) {
51. **for** (**int** j = 0; j < 366; j++) {
52. Days[i][j][0] = getMonth(i + START\_YEAR, j);
53. Days[i][j][1] = getDay(i + START\_YEAR, j);
54. Days[i][j][2] = getWeekSeq0fYear(i + START\_YEAR, Days[i][j][0], Days[i][j][1]);
55. Days[i][j][3] = getDaySeq0fWeek(i + START\_YEAR, j);
56. }
57. }
58. }
60. // 查找某人生日并打印周历的函数
61. **void** findBirthday(**int** year, **int** month, **int** day, **int**\* weekend) {
62. **int** isExist = 0;
63. // 遍历Days数组，查找指定年月日
64. **for** (**int** i = 0; i < 366; i++) {
65. **if** (Days[year - START\_YEAR][i][0] == month && Days[year - START\_YEAR][i][1] == day) {
66. // 找到生日，打印周历
67. isExist = 1;
68. **if** (Days[year - START\_YEAR][i][3] == 5 || Days[year - START\_YEAR][i][3] == 6) {
69. (\*weekend)++;
70. }
71. printWeekCalendar(year, Days[year - START\_YEAR][i][2]);
72. **break**;
73. }
74. }
75. **if** (isExist == 0) {
76. printf("No such birthday found in %d .\n",year);
77. }
78. }
80. // 打印周历的函数
81. **void** printWeekCalendar(**int** year, **int** weekSeq) {
82. printf("Year: %d, Week %d Calendar:\n", year, weekSeq);
83. printf(" #M:%10s%10s%10s%10s%10s%10s%10s\n","Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
84. printOneWeek(year, weekSeq);
85. }
86. **int** isLeapYear(**int** year) {
87. **return** (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
88. }
89. **int** getDaySeqOnJan1(**int** year) {
90. **int** totalDays = 0;
91. **for** (**int** i = 1900; i < year; i++) {
92. totalDays += isLeapYear(i) ? 366 : 365;
93. }
94. **return** totalDays % 7; // 0: 周日, 1: 周一, ..., 6: 周六
95. }
96. **int** getMonthLength(**int** year, **int** month) {
97. **if** (month == 2) {
98. **return** isLeapYear(year) ? 29 : 28;
99. }
100. **else** **if** (month == 4 || month == 6 || month == 9 || month == 11) {
101. **return** 30;
102. }
103. **else** {
104. **return** 31;
105. }
106. }
107. **int** getDaySeq(**int** year, **int** month, **int** day) {
108. **int** totalDays = 0;
109. **for** (**int** i = 1; i < month; i++) {
110. totalDays += getMonthLength(year, i);
111. }
112. totalDays += day;
113. **return** totalDays;
114. }
115. **int** getWeekSeq0fYear(**int** year, **int** month, **int** day) {
116. **return** (getDaySeqOnJan1(year) + getDaySeq(year, month, day) - 1) / 7 + 1;
117. }
118. **int** getMonth(**int** year, **int** daySeq0fYear) {
119. **int** month = 1;
120. **while** (daySeq0fYear > getMonthLength(year, month)) {
121. daySeq0fYear -= getMonthLength(year, month);
122. month++;
123. }
124. **return** month;
125. }
126. **int** getDay(**int** year, **int** daySeq0fYear) {
127. **int** month = getMonth(year, daySeq0fYear);
128. **return** daySeq0fYear - getDaySeq(year, month, 1) + 1;
129. }
130. **int** getDaySeq0fWeek(**int** year, **int** daySeq0fYear) {
131. **return** (getDaySeqOnJan1(year) + daySeq0fYear - 1) % 7;
132. }
133. **int** getNextMonday(**int** year, **int** daySeq0fYear) {
134. **int** daySeq0fWeek = getDaySeq0fWeek(year, daySeq0fYear);
135. **return** daySeq0fYear + (daySeq0fWeek == 0 ? 1 : 8 - daySeq0fWeek);
136. }
137. **int** getThisMonday(**int** year, **int** daySeq0fYear) {
138. **int** daySeq0fWeek = getDaySeq0fWeek(year, daySeq0fYear);
139. **return** daySeq0fYear - (daySeq0fWeek == 0 ? 6 : daySeq0fWeek - 1);
140. }
141. **int** getThisSunday(**int** year, **int** daySeq0fYear) {
142. **int** daySeq0fWeek = getDaySeq0fWeek(year, daySeq0fYear);
143. **return** daySeq0fYear + (daySeq0fWeek == 0 ? 0 : 7 - daySeq0fWeek);
144. }
146. **void** printOneDay(**int** year, **int** daySeq0fYear) {
147. **if** ( daySeq0fYear == 1) {
148. **if** (getDaySeq0fWeek(year, daySeq0fYear) == 5 || getDaySeq0fWeek(year, daySeq0fYear) == 6) {
149. printf(" %02d.%02d.%02d!", year - 2000, getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
150. }
151. **else** {
152. printf("  %02d.%02d.%02d", year - 2000, getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
153. }
154. }
155. **else** **if** (getDay(year, daySeq0fYear) == 1) {
156. **if** (getDaySeq0fWeek(year, daySeq0fYear) == 5 || getDaySeq0fWeek(year, daySeq0fYear) == 6) {
157. printf("    %02d.%02d!", getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
158. }
159. **else** {
160. printf("     %02d.%02d", getMonth(year, daySeq0fYear), getDay(year, daySeq0fYear));
161. }
162. }
163. **else** **if** (getDaySeq0fWeek(year, daySeq0fYear) == 5 || getDaySeq0fWeek(year, daySeq0fYear) == 6) {
164. printf("%7s%02d!", " ", getDay(year, daySeq0fYear));
165. }
166. **else** {
167. printf("%8s%02d", " ", getDay(year, daySeq0fYear));
168. }
169. }
171. **void** printOneWeek(**int** year, **int** weekSeq0fYear) {
173. // Before Monday, print the week sequence
174. printf("[%02d]", weekSeq0fYear);
176. // Declare variables to navigate the semester
177. **int** sStartSeq0fYear = 7 \* (weekSeq0fYear - 1) - getDaySeqOnJan1(year) + 1;
178. **int** sEndSeq0fYear = sStartSeq0fYear + 6;
179. **int** currentYearLength = (isLeapYear(year) == 1) ? 366 : 365;
180. **int** daySeq0fYear = sStartSeq0fYear;
182. // If this is the first week of semester, show full month and day for first da
183. // Print the semester calendar
184. **for** (; daySeq0fYear <= sEndSeq0fYear; daySeq0fYear++) {
186. // Print the month and day for the current day
187. **if** (daySeq0fYear <= currentYearLength) {
188. printOneDay(year, daySeq0fYear);
190. }
191. **else** {
192. printOneDay(year + 1, daySeq0fYear - currentYearLength);
193. }
195. }
196. // After Sunday, print a new line
197. printf("\n");
198. }
199. 实验三

（1）main.c

1. #include <stdio.h>
2. #define YEAR\_NUM 3 // 假设我们有3年的数据
4. **typedef** **struct** {
5. **int** year;
6. **int** daySeq;
7. **int** month;
8. **int** day;
9. **int** weekSeq;
10. **int** weekDay;
11. } Day;
13. **typedef** **enum** {
14. DATE\_INFO\_FULL,
15. DATE\_INFO\_BRIEF,
16. } DateFormat;
18. **int** getDaySeqOnJan1(**int** year);
20. // Functions on different year
21. **int** isLeapYear(**int** year);
23. // Functions about month and day
24. **int** getMonthLength(**int** year, **int** month);
25. **int** getDaySeq(**int** year, **int** month, **int** day);
26. **int** getWeekSeq0fYear(**int** year, **int** month, **int** day);
28. // Functions for properties of one day
29. **int** getMonth(**int** year, **int** daySeqOfYear);
30. **int** getDay(**int** year, **int** daySeqOfYear);
31. **int** getDaySeqOfWeek(**int** year, **int** daySeq0fYear);
33. // Functions support multiple year
34. **void** setYearArray(Day yearArray[][366], **int** yearNum, **int** yearStart);
36. // Functions support struct Day
37. **int** isDay(Day year[], **int** month, **int** day);
38. Day getDayBefore(Day currentDay, **int** interval);
39. Day getDayAfter(Day currentDay, **int** interval);
40. **void** printDay(Day currentDay, **int** displayFormat);
41. **int** main() {
42. Day Years[YEAR\_NUM][366] = { 0 }; // 假设的年份数组
43. **int** inputMonth, inputDay,inputyear;
44. setYearArray(Years, YEAR\_NUM, 2023); // 假设的年份从2023年开始
46. // 用户输入生日的月份和日期
47. printf("Enter birth month and day: ");
48. scanf\_s("%d %d", &inputMonth, &inputDay);
49. printf("enter the year you want to check (2023-2025): ");
50. scanf\_s("%d", &inputyear);
51. **if** (inputyear < 2023 || inputyear > 2025) {
52. printf("Invalid year.\n");
53. **return** 0;
54. }
55. **int** day = isDay(Years[inputyear - 2023], inputMonth, inputDay);
56. **if** (day == -1) {
57. printf("Invalid date.\n");
58. **return** 0;
59. }
61. printf(" #:%10s%10s%10s%10s%10s%10s%10s", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");
62. Day birthday = Years[inputyear - 2023][day];
63. Day startday = getDayBefore(birthday, 2);
64. printDay(startday, DATE\_INFO\_FULL);
65. Day secondday = getDayBefore(birthday, 1);
66. **if** (secondday.weekDay == 0) {
67. printDay(secondday, DATE\_INFO\_FULL);
68. }
69. **else** {
70. (printDay(secondday, DATE\_INFO\_BRIEF));
71. }
72. **if** (birthday.weekDay == 0) {
73. printDay(birthday, DATE\_INFO\_FULL);
74. }
75. **else** {
76. printDay(birthday, DATE\_INFO\_BRIEF);
77. }
78. **return** 0;
79. }
81. **int** getDaySeq(**int** year, **int** month, **int** day) {
82. **int** totalDays = 0;
83. **for** (**int** i = 1; i < month; i++) {
84. totalDays += getMonthLength(year, i);
85. }
86. totalDays += day;
87. **return** totalDays;
88. }
89. **int** getMonthLength(**int** year, **int** month) {
90. **if** (month == 2) {
91. **return** isLeapYear(year) ? 29 : 28;
92. }
93. **else** **if** (month == 4 || month == 6 || month == 9 || month == 11) {
94. **return** 30;
95. }
96. **else** {
97. **return** 31;
98. }
99. }
100. **int** getWeekSeq0fYear(**int** year, **int** month, **int** day) {
101. **return** (getDaySeqOnJan1(year) + getDaySeq(year, month, day) - 1) / 7 + 1;
102. }
103. **int** isLeapYear(**int** year) {
104. **return** (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
105. }
106. **int** getDaySeqOnJan1(**int** year) {
107. **int** totalDays = 0;
108. **for** (**int** i = 1900; i < year; i++) {
109. totalDays += isLeapYear(i) ? 366 : 365;
110. }
111. **return** totalDays % 7; // 0: 周日, 1: 周一, ..., 6: 周六
112. }
113. **int** getMonth(**int** year, **int** daySeq0fYear) {
114. **int** month = 1;
115. **while** (daySeq0fYear > getMonthLength(year, month)) {
116. daySeq0fYear -= getMonthLength(year, month);
117. month++;
118. }
119. **return** month;
120. }
121. **int** getDay(**int** year, **int** daySeq0fYear) {
122. **int** month = getMonth(year, daySeq0fYear);
123. **return** daySeq0fYear - getDaySeq(year, month, 1) + 1;
124. }
125. **int** getDaySeqOfWeek(**int** year, **int** daySeq0fYear) {
126. **return** (getDaySeqOnJan1(year) + daySeq0fYear - 1) % 7;
127. }
128. **void** setYearArray(Day yearArray[][366], **int** yearNum, **int** yearStart) {
129. **for** (**int** i = 0; i < yearNum; i++) {
130. **for** (**int** j = 0; j < 366; j++) {
131. yearArray[i][j].year = yearStart + i;
132. yearArray[i][j].daySeq = j;
133. yearArray[i][j].month = getMonth(yearArray[i][j].year, yearArray[i][j].daySeq);
134. yearArray[i][j].day = getDay(yearArray[i][j].year, yearArray[i][j].daySeq);
135. yearArray[i][j].weekSeq = getWeekSeq0fYear(yearArray[i][j].year, yearArray[i][j].month, yearArray[i][j].day);
136. yearArray[i][j].weekDay = getDaySeqOfWeek(yearArray[i][j].year, yearArray[i][j].daySeq);
138. }
139. }
140. }
141. **int** isDay(Day year[], **int** month, **int** day) {
142. **for** (**int** i = 0; i < 366; i++) {
143. **if** (year[i].month == month && year[i].day == day) {
144. **return** i;
145. }
146. }
147. **return** -1;
148. }
149. Day getDayBefore(Day currentDay, **int** interval) {
150. Day newDay = currentDay;
151. newDay.daySeq -= interval;
152. **if** (newDay.daySeq < 0) {
153. newDay.year--;
154. newDay.daySeq += isLeapYear(newDay.year) ? 366 : 365;
155. }
156. newDay.month = getMonth(newDay.year, newDay.daySeq);
157. newDay.day = getDay(newDay.year, newDay.daySeq);
158. newDay.weekSeq = getWeekSeq0fYear(newDay.year, newDay.month, newDay.day);
159. newDay.weekDay = getDaySeqOfWeek(newDay.year, newDay.daySeq);
160. **return** newDay;
161. }
162. Day getDayAfter(Day currentDay, **int** interval) {
163. Day newDay = currentDay;
164. newDay.daySeq += interval;
165. **if** (newDay.daySeq >= (isLeapYear(newDay.year) ? 366 : 365)) {
166. newDay.year++;
167. newDay.daySeq -= isLeapYear(newDay.year) ? 366 : 365;
168. }
169. newDay.month = getMonth(newDay.year, newDay.daySeq);
170. newDay.day = getDay(newDay.year, newDay.daySeq);
171. newDay.weekSeq = getWeekSeq0fYear(newDay.year, newDay.month, newDay.day);
172. newDay.weekDay = getDaySeqOfWeek(newDay.year, newDay.daySeq);
173. **return** newDay;
174. }
175. **void** printDay(Day currentDay, **int** displayFormat) {
176. **if** (displayFormat == DATE\_INFO\_FULL) {
177. printf("\n#%02d:", currentDay.weekSeq);
178. **for** (**int** i = 0; i < currentDay.weekDay; i++) {
179. printf("%10s", " ");
180. }
181. printf("%2s%02d.%02d.%02d", " ", currentDay.year - 2000, currentDay.month, currentDay.day);
182. }
183. **else** **if** (displayFormat == DATE\_INFO\_BRIEF) {
184. printf("%2s%02d.%02d.%02d", " ", currentDay.year - 2000, currentDay.month, currentDay.day);
185. }
186. }
187. 实验四

（1）main.c

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
5. **typedef** **struct** Date{
6. **int** year;
7. **int** month;
8. **int** day;
9. **int** weekday;
10. **int** weeknum;
11. **struct** Date\* prev;
12. **struct** Date\* next;
13. } Date;
14. **const** **int** NormalDayOfMonth[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
16. **const** **int** LeapDayOfMonth[12] = { 31,29,31,30,31,30,31,31,30,31,30,31 };
18. **int** is\_leap\_year(**int** year);
20. **void** create\_date(**int** start\_year, **int** end\_year, **int** start\_weekday, Date\*\* head);
22. **void** find\_date(Date\* head, **int** month, **int** day, **int** year);
24. **void** destroy\_date(Date\* head);
26. **int** main() {
27. printf("enter the birthday(for instance 2 13):");
28. **int** month, day;
29. scanf\_s("%d %d", &month, &day);
30. printf("enter the start year and end year you want to find the birthday and the two days before it(for instance 2019 2025):");
31. **int** start\_year, end\_year;
32. scanf\_s("%d %d", &start\_year, &end\_year);
33. start\_year -= 1;//从前一年开始，以防前两天的日期超出范围
34. printf("enter the start weekday of the first day of %d (2 mean Tues,0 mean Sun):",start\_year);
35. **int** start\_weekday;
36. scanf\_s("%d", &start\_weekday);
37. Date\* head = NULL;
38. create\_date(start\_year, end\_year, start\_weekday, &head);
39. //利用while循环可以多次输入找到不同年份的生日，输入0结束
40. **while** (1) {
41. printf("enter the year you want to search(enter 0 if you want to end):");
42. **int** year;
43. scanf\_s("%d",&year);
44. **if** (year != 0) {
45. find\_date(head, month, day, year);
46. printf("\n");
47. }
48. **else** {
49. **break**;
50. }
51. }
52. destroy\_date(head);
53. **return** 0;
54. }
56. **int** is\_leap\_year(**int** year) {
57. **if** (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {
58. **return** 1;
59. }
60. **else** {
61. **return** 0;
62. }
63. }
64. **void** create\_date(**int** start\_year, **int** end\_year, **int** start\_weekday,Date\*\* head) {
65. Date\* prev = NULL;
66. **int** weekday = start\_weekday;
67. **for** (**int** year = start\_year; year <= end\_year; year++) {
68. **int** weeknum = 1;
69. **for** (**int** month = 1; month <= 12; month++) {
70. **int** days;
71. **if** (is\_leap\_year(year)) {
72. days = LeapDayOfMonth[month - 1];
73. }
74. **else** {
75. days = NormalDayOfMonth[month - 1];
76. }
77. **for** (**int** day = 1; day <= days; day++) {
78. Date\* cur = (Date\*)malloc(**sizeof**(Date));
79. **if** (prev == NULL) {
80. \*head = cur;
81. }
82. **else** {
83. prev->next = cur;
84. }
85. cur->year = year;
86. cur->month = month;
87. cur->day = day;
88. cur->weekday = weekday;
89. cur->weeknum = weeknum;
90. cur->prev = prev;
91. prev = cur;
92. weekday = (weekday + 1) % 7;
93. **if** (weekday == 1) {
94. weeknum++;
95. }
96. }
97. }
98. }
99. prev->next = NULL;
100. }
102. **void** find\_date(Date\* head, **int** month, **int** day, **int** year) {
103. Date\* cur = head;
104. **while** (cur != NULL) {
105. **if** (cur->year == year && cur->month == month && cur->day == day) {
106. Date\* pre1 = cur->prev;
107. Date\* pre2 = pre1->prev;
108. printf("%d.%d.%d is %dth day of %dth week\n", year, month, day, cur->weekday, cur->weeknum);
109. printf("#W:%10s%10s%10s%10s%10s%10s%10s\n", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.","Sun.");
110. **int** week = pre2->weeknum;
111. printf("%02d:", week);
112. **if** (pre2->weekday == 0) {
113. printf("%60s", " ");
114. }
115. **else** {
116. **for** (**int** i = 0; i < pre2->weekday-1; i++) {
117. printf("%10s", " ");
118. }
119. }
120. printf("%2s%02d.%02d.%02d"," ",pre2->year-2000, pre2->month, pre2->day);
121. **if** (pre1->weekday == 1) {
122. printf("\n%02d:", pre1->weeknum);
123. }
124. printf("%2s%02d.%02d.%02d", " ", pre1->year - 2000, pre1->month, pre1->day);
125. **if** (cur->weekday == 1) {
126. printf("\n%02d:", cur->weeknum);
127. }
128. printf("%2s%02d.%02d.%02d", " ", cur->year - 2000, cur->month, cur->day);
129. **break**;
130. }
131. cur = cur->next;
132. }
133. **if** (cur == NULL) {
134. printf("There birthday do not exist\n");
135. }
136. }
138. **void** destroy\_date(Date\* head) {
139. Date\* cur = head;
140. **while** (cur != NULL) {
141. Date\* next = cur->next;
142. free(cur);
143. cur = next;
144. }
145. }
146. 实验五
147. main.cpp
148. #include <iostream>
149. #include <list>
150. #include <sstream>
151. #include "Date.h"
152. **using** **namespace** std;

155. // 读取日期字符串并解析为 Date 对象
156. Date parseDate(**const** string& dateStr);
157. **int** daysBetween(**const** Date& date1, **const** Date& date2);
158. **int** getDayOfWeek(**const** Date& date);
159. **int** main() {
160. string startDateStr, endDateStr;
161. cout << "Enter start date (YYYY.MM.DD): ";
162. cin >> startDateStr;
163. cout << "Enter end date (YYYY.MM.DD): ";
164. cin >> endDateStr;
166. Date startDate = parseDate(startDateStr);
167. Date endDate = parseDate(endDateStr);
169. // 创建一个链表来存储不同天数的日期
170. list<Date> dates;
172. // 生成中间的所有日期
173. Date currentDate = startDate;
174. **while** (currentDate < endDate + 1) {
175. dates.push\_back(currentDate);
176. currentDate = currentDate + 1; // 日期加一天
177. }
179. // 打印所有日期
180. **int** week=1,startWeekday=getDayOfWeek(dates.front());
181. **if** (startWeekday == 0) {
182. startWeekday = 7;
183. }
184. cout << "Dates between " << startDate << " and " << endDate << ":\n";
185. cout << " ##:" << setw(10)<<setfill(' ') << "Mon." << setw(10) << "Tues."<< setw(10) << "Wed." << setw(10) << "Thur."<< setw(10) << "Fri." << setw(10) << "Sat."<< setw(10) << "Sun." << endl;
186. cout << "001:  ";
187. **for** (**int** i = 1; i < startWeekday; i++) {
188. cout << setw(10) << " ";
189. }
191. **for** (**const** Date& date : dates) {
192. cout  << date<< setw(2)<<setfill(' ') << " ";
193. **if** (getDayOfWeek(date) == 0) {
194. week++;
195. cout <<endl<< setw(3) << setfill('0') << week << ":"<<setw(2) << setfill(' ') << " ";
196. }
197. }
199. **return** 0;
200. }
201. Date parseDate(**const** string& dateStr) {
202. istringstream iss(dateStr);
203. **int** y, m, d;
204. **char** dot1, dot2;
205. iss >> y >> dot1 >> m >> dot2 >> d;
206. **return** Date(y, m, d);
207. }
209. **int** daysBetween(**const** Date& date1, **const** Date& date2) {
210. **int** daysInMonth[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
211. **int** days = 0;
213. // 计算年份差对应的天数
214. **for** (**int** year = date1.year; year < date2.year; ++year) {
215. days += 365;
216. **if** ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
217. days += 1;  // 闰年多一天
218. }
219. }
221. // 计算月份差对应的天数
222. **for** (**int** month = date1.month; month < date2.month; ++month) {
223. days += daysInMonth[month - 1];
224. **if** (month == 2 && ((date2.year % 4 == 0 && date2.year % 100 != 0) || (date2.year % 400 == 0))) {
225. days += 1;  // 2月在闰年多一天
226. }
227. }
229. // 计算日差对应的天数
230. days += date2.day - date1.day;
232. **return** days;
233. }
234. **int** getDayOfWeek(**const** Date& date) {
235. Date baseDate = { 2000, 1, 1 };  // 已知2000年1月1日是星期六
236. **int** daysDiff = daysBetween(baseDate, date);
237. **return** (daysDiff + 6) % 7;
238. }
239. Date.h
240. #pragma once
242. #include <iostream>
243. #include <iomanip>

246. **class** Date {
247. **public**:
248. **int** year;
249. **int** month;
250. **int** day;
251. // 构造函数
252. Date(**int** y, **int** m, **int** d);
253. // 重载加法运算符
254. Date operator+(**int** days) **const**;
255. // 重载输出流运算符
256. **friend** std::ostream& operator<<(std::ostream& os, **const** Date& date);
257. // 重载小于运算符
258. **bool** operator<(**const** Date& other) **const**;
259. };
260. Date.cpp
261. #include "Date.h"
262. #include <sstream>
264. // 构造函数
265. Date::Date(**int** y, **int** m, **int** d) : year(y), month(m), day(d) {}
266. // 重载加法运算符
267. Date Date::operator+(**int** days) **const** {
268. Date result = \***this**; // 创建一个当前日期的副本
269. result.day += days; // 加上指定的天数
270. **int** daysInMonth[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
271. **if** (result.month == 2 && (result.year % 4 == 0 && (result.year % 100 != 0 || result.year % 400 == 0))) {
272. daysInMonth[1] = 29; // 闰年2月有29天
273. }
274. **while** (result.day > daysInMonth[result.month - 1]) {
275. result.day -= daysInMonth[result.month - 1]; // 减去当前月的天数
276. result.month++;
278. **if** (result.month > 12) {
279. result.month = 1;
280. result.year++;
281. }
282. }
284. **return** result;
285. }
286. // 重载输出流运算符
287. std::ostream& operator<<(std::ostream& os, **const** Date& date) {
288. os << std::setw(2) << std::setfill('0') << date.year-2000 << "." << std::setw(2) << std::setfill('0') << date.month << "." << std::setw(2) << std::setfill('0') << date.day;
289. **return** os;
290. }
291. // 重载小于运算符
292. **bool** Date::operator<(**const** Date& other) **const** {
293. **if** (year != other.year) **return** year < other.year;
294. **if** (month != other.month) **return** month < other.month;
295. **return** day < other.day;
296. }

以上是我的实验报告，祝刘老师过个好年。