

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 | 陈泽州 | 学号 | U202411198 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2024.12 | 地点 | 华中科技大学 |

|  |  |  |  |
| --- | --- | --- | --- |
| 成绩 |  | 教师 | 刘威 |

# 实验目的

完成日历系列代码（日历系列、大数计算系列，选择其中之一）。

# 实验环境

操作系统：Windows 10

编程工具：CodeBlocks 16.01

# 实验一

## 实验任务

1. 计算周几
2. 打印周历
3. 计算过了多少天

## 实验步骤

图示

描述已自动生成

## 代码测试

### 输出测试结果

日历

描述已自动生成

### 2月29日

手机屏幕的截图

描述已自动生成

### 跨年份测试

图片包含 日历

描述已自动生成

日历

描述已自动生成

## 实验结论

是

## 实验总结

此实验有助于我熟练运用结构体，以及printf的用法，以及对函数的使用，以及工程有了更深刻的认识

# 本课程学习总结

## 一、初识C语言：从C++到C的转变

在初高中时期，我接触到了C++，那时的我被它强大的功能和灵活的特性所吸引，逐渐熟悉了面向对象编程的思维方式，也养成了自己独特的代码风格。然而，当真正开始学习C语言时，才发现这并非简单的“换汤不换药”。

C语言与C++有着本质的区别，它更加注重底层操作和对硬件的直接控制。一开始，我总是不自觉地将C++的思维模式和代码习惯带入到C语言的学习中。比如，在C++中常用的sort函数，在C语言中却无法直接使用，这让我感到非常不适应。每次遇到需要排序的情况，我都得重新思考如何用C语言的方式来实现，这无疑增加了学习的难度。

记得有一次，我在编写一个简单的数据排序程序时，习惯性地想要调用sort函数，结果程序编译报错。我这才意识到自己又犯了“老毛病”。为了克服这个困难，我开始深入研究C语言的排序算法，从冒泡排序到快速排序，一步步地去理解它们的原理和实现过程。通过不断地练习和实践，我逐渐习惯了C语言的编程方式，也慢慢学会了如何用C语言的思维去解决问题。

## 二、逻辑困境：思维的碰撞与突破

学习C语言的过程中，逻辑问题一直是我最大的“拦路虎”。

## 三、代码规范与调试技巧：从懵懂到熟练

在学习C++的初期，我对代码规范和调试技巧几乎一无所知。我写的代码常常是杂乱无章，既没有合理的缩进，也没有清晰的注释，甚至连变量命名都是随意取的。这不仅让别人难以理解我的代码，也给自己后续的维护和调试带来了很大的困难。

随着学习的深入，我逐渐意识到了代码规范的重要性。我开始学习并遵循一些基本的代码规范，如使用有意义的变量名、合理地进行代码缩进、添加必要的注释等。这些小小的改变，让我的代码变得清晰易懂，也提高了我的编程效率。

## 四、总结与展望

回顾学习C语言的这段历程，虽然充满了心酸和挫折，但也让我收获了成长和进步。从最初对C语言的陌生和困惑，到逐渐熟悉它的语法和逻辑；从习惯性地使用C++的代码风格，到慢慢适应C语言的编程方式；从对代码规范和调试技巧的懵懂，到如今能够熟练地运用它们，这一路走来，我付出了很多努力，也收获了很多宝贵的经验。

学习C语言不仅让我掌握了一门新的编程语言，更让我学会了如何思考问题、如何解决问题。在未来的日子里，我将继续深入学习C语言，不断提升自己的编程能力。我相信，只要我坚持不懈，勇于面对困难和挑战，就一定能够在编程的道路上走得更远，实现自己的梦想。

# 附录

完整实验代码附在此处

1. xxx实验
2. main.c
3. #include<stdio.h>
4. #include<math.h>
5. #include "func.h"
6. #define YEAR\_NUM           9
7. #define YEAR\_MIN           2020
8. #define JAN1INYEARMIN      3
9. #define DATE\_INFO\_BRIEF    1
10. #define DATE\_INFO\_FULL     2
11. #define DATE\_NOSHOW        3
12. #define DATE\_STAR          4
14. **const** **static** **int** Month[2][13] = { {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31} , {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31} };
15. //0 for normal ; 1 for leap ;
17. **typedef** **struct** DAY\_ Day;
19. **int** main(**void**)
20. {
21. **int** i, inputMonth, inputDay, displayFormat;
22. Day birthday, prepareday,printStartDay,currentDay;
23. **int** printDayRange = 3;
24. **int** Years[YEAR\_NUM];
26. setYearArray(Years, YEAR\_NUM, YEAR\_MIN);
28. printf("Finding Birthday in year (%d-%d), please input the month and day : ", YEAR\_MIN, YEAR\_MIN + YEAR\_NUM - 1 );
29. scanf("%d%d", &inputMonth, &inputDay);
31. **if** (inputMonth < 1 || inputMonth > 12)
32. {
33. printf("Sorry, the input month are invalid.\n");
34. **return** 1;
35. }**else** **if**( inputDay < 1 || inputDay > Month[1][inputMonth] )
36. {
37. printf("Sorry, the input day are invalid.\n");
38. **return** 1;
39. }
41. **for** ( i = 0 ; i < YEAR\_NUM ; i ++ )
42. {
43. printf( "\n%s%s%d\n", " ", "Birthday in Year ", Years[i] );
44. **if** ( !isDay(Years[i], inputMonth, inputDay) )
45. {
46. printf(" Not found.\n");
47. **continue**;
48. }
49. birthday = setDay ( Years[i], inputMonth, inputDay );
50. prepareday = getDayBefore( birthday, printDayRange );
51. printStartDay = getDayBefore( prepareday, prepareday.weekDay - 1 );
52. printf(" #W:%10s%10s%10s%10s%10s%10s%10s\n","Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.");
53. **for** ( currentDay = printStartDay; getTwoDaysInterval(currentDay, birthday) <= 0;
54. currentDay = getDayAfter( currentDay, 1 ) )
55. {
56. **if** ( getTwoDaysInterval(currentDay, prepareday) < 0 )
57. {
58. displayFormat = DATE\_NOSHOW;
59. }
60. **else** **if** ( getTwoDaysInterval(currentDay, prepareday) == 0 || (currentDay.month == 1 && currentDay.day == 1))
61. {
62. displayFormat = DATE\_INFO\_FULL;
63. }
64. **else** **if** ( getTwoDaysInterval(currentDay, birthday) < 0 )
65. {
66. displayFormat = DATE\_INFO\_BRIEF;
67. }
68. **else** **if** ( getTwoDaysInterval(currentDay, birthday) == 0 )
69. {
70. displayFormat = DATE\_STAR;
71. }
72. printDay(currentDay, displayFormat);
73. }
74. printf("\n");
75. }
76. }

（2）Func.h

1. #ifndef FUNC\_H
2. #define FUNC\_H
4. **struct** DAY\_
5. {
6. **int** year;
7. **int** month;
8. **int** day;
9. **int** daySeq;
10. **int** weekSeq;
11. **int** weekDay;
12. };
14. **typedef** **struct** DAY\_ Day;
16. **int** isLeapyear( **int** x );
18. **int** getDaySeq( **int** year , **int** month , **int** day );
19. **int** isLeapYear(**int** year);
20. **int** getDaySeqOnJan1(**int** year);
22. // Functions about month and day
23. **int** getDaySeq(**int** year, **int** month, **int** day);
24. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day);
26. // Functions for properties of one day
27. **int** getMonth(**int** year, **int** daySeqOfYear);
28. **int** getDay(**int** year, **int** daySeqOfYear);
29. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear);
31. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin);
33. **int** isDay(**int** year, **int** month, **int** day);
34. Day setDay(**int** year, **int** month, **int** day);
35. Day getDayBefore(Day currentday, **int** interval);
36. Day getDayAfter(Day currentday, **int** interval);
37. **int** getTwoDaysInterval(Day startDay, Day endDay);
38. **void** printDay(Day currentday, **int** displayFormat);
40. #endif

（3）Func.c

1. #include "func.h"
2. #include <stdio.h>
3. #include <math.h>
5. #define YEAR\_NUM           9
6. #define YEAR\_MIN           2020
7. #define JAN1INYEARMIN      3
8. #define DATE\_INFO\_BRIEF    1
9. #define DATE\_INFO\_FULL     2
10. #define DATE\_NOSHOW        3
11. #define DATE\_STAR          4
13. **typedef** **struct** DAY\_ Day;
15. **const** **static** **int** Month[2][13] = { {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31} , {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31} };
16. //0 for normal ; 1 for leap ;
18. **int** isLeapYear(**int** year)
19. {
20. **return** year % 4 == 0 && ( year % 100 != 0 || year % 400 == 0 ) ;
21. }
23. **int** getDaySeqOnJan1(**int** year)
24. {
25. **int** day = JAN1INYEARMIN;
26. **int** i;
27. **for**( i = YEAR\_MIN ; i < year ; i++ )
28. {
29. day = ( day + isLeapYear(i) ) % 7 + 1 ;
30. }
31. **return** day;
32. }
34. **int** getDaySeq(**int** year, **int** month, **int** day)
35. {
36. **int** IsLeap = isLeapYear(year);
37. **do**
38. {
39. day += Month[IsLeap][--month];
40. } **while** ( month > 1 );
41. **return** day;
42. }
44. **int** getWeekSeqOfYear(**int** year, **int** month, **int** day)
45. {
46. **int** daySeqOfYear = getDaySeq(year, month, day);
47. daySeqOfYear = daySeqOfYear + getDaySeqOnJan1(year) - 1;
48. **int** week;
49. week = daySeqOfYear / 7 + 1;
50. **return** week;
51. }
53. **int** getMonth(**int** year, **int** daySeqOfYear)
54. {
55. **int** month ;
56. **int** isLeap = isLeapYear(year);
57. **for**(month = 1; daySeqOfYear > Month[isLeap][month]; month ++)
58. {
59. daySeqOfYear -= Month[isLeap][month];
60. }
61. **return** month;
62. }
63. **int** getDay(**int** year, **int** daySeqOfYear)
64. {
65. **int** month ;
66. **int** isLeap = isLeapYear(year);
67. **for**(month = 1;daySeqOfYear > Month[isLeap][month]; month ++)
68. {
69. daySeqOfYear -= Month[isLeap][month];
70. }
71. **return** daySeqOfYear;
72. }
73. **int** getDaySeqOfWeek(**int** year, **int** daySeqOfYear)
74. {
75. **int** day = (daySeqOfYear + getDaySeqOnJan1(year) - 2) % 7 + 1 ;
76. **return** day;
77. }
79. **void** setYearArray(**int** Years[], **int** yearNum, **int** yearMin)
80. {
81. **int** i = 0;
82. **for**(i = 0; i < yearNum; i++)
83. {
84. Years[i] = yearMin + i;
85. }
87. **return** ;
88. }
90. **int** isDay(**int** year, **int** month, **int** day)
91. {
92. **return** day <= Month[isLeapYear(year)][month];
93. }
94. Day setDay(**int** year, **int** month, **int** day)
95. {
96. Day d;
97. d.year = year;
98. d.month = month;
99. d.day = day;
100. d.daySeq = getDaySeq(year, month, day);
101. d.weekSeq = getWeekSeqOfYear(year, month, day);
102. d.weekDay = getDaySeqOfWeek(year, d.daySeq);
103. **return** d;
104. }
105. Day getDayBefore(Day currentday, **int** interval)
106. {
107. **while** (currentday.day <= interval)
108. {
109. interval -= currentday.day;
111. **if**( currentday.month == 1 )
112. {
113. currentday.day = 31;
114. currentday.year --;
115. currentday.month = 12;
116. }**else**
117. {
118. currentday.month--;
119. currentday.day = Month[isLeapYear(currentday.year)][currentday.month];
120. }
121. }
122. currentday.day -= interval;
123. **return** setDay(currentday.year,currentday.month,currentday.day);
124. }
125. Day getDayAfter(Day currentday, **int** interval)
126. {
127. **while**( interval > 0 )
128. {
129. **if**( currentday.day + interval > Month[isLeapYear(currentday.year)][currentday.month] )
130. {
131. currentday.day = 1;
132. interval -= Month[isLeapYear(currentday.year)][currentday.month] - currentday.day + 1;
133. currentday.month ++;
134. **if**( currentday.month > 12 )
135. {
136. currentday.month = 1;
137. currentday.year ++;
138. }
139. }
140. **else**
141. {
142. currentday.day += interval;
143. interval = 0;
144. }
145. }
146. **return** setDay(currentday.year,currentday.month,currentday.day);
147. }
149. **int** getTwoDaysInterval(Day startDay, Day endDay)
150. {
151. **while**( startDay.year != endDay.year )
152. {
153. **if**( startDay.year > endDay.year )
154. {
155. startDay.year --;
156. startDay.daySeq += 365 + isLeapYear(startDay.year);
157. }**else**
158. {
159. endDay.year--;
160. endDay.daySeq += 365 + isLeapYear(endDay.year);
161. }
162. }
163. **return** startDay.daySeq - endDay.daySeq;
164. }
166. **void** printDay(Day currentday, **int** displayFormat)
167. {
168. **if**( currentday.weekDay == 1 )
169. {
170. printf("\n[%02d]" , currentday.weekSeq);
171. }
172. **switch** (displayFormat)
173. {
174. **case** 1:
175. printf("%10d", currentday.day);
176. **break**;
177. **case** 2:
178. printf("%4d.%02d.%02d", (currentday.year % 100), currentday.month, currentday.day);
179. **break**;
180. **case** 3:
181. printf("%10s", "");
182. **break**;
183. **case** 4:
184. printf("    %02d.%02d\*", currentday.month, currentday.day);
185. **break**;
186. }
188. **return**;
189. }