

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 | 刘欣 | 学号 | U202410809 |

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

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

# 实验目的

完成日历系列系列代码。

# 实验环境

操作系统：Windows 10

编程工具：CodeBlocks 13.12

# 实验一(日历6)

## 实验任务

[实验目标]

巩固函数的使用，对已有函数进行功能升级，支持跨年的打印功能，体会代码模块化给代码维护带来的优势

[实验任务]

1.用函数改写打印华中科技大学校历的程序，打印2021年秋季学期的校历

2.秋季学期从9月1日所在周的周一开始，到第二年1月的某日结束：用户输入1月份的某日作为参考日，取该日所在周周日作为学期的结束。

3.更新相关函数，支持不同的年份的日期，计算相关函数需要增加一个参数int year

[要求函数]

1. int isLeapYear(int year);

2. int getDaySeqOnJan1(int year);

3. int getMonthLength(int year, int month);

4. int getDaySeq(int year, int month, int day);

5. int getMonth(int year, int daySeqOfYear);

6. int getDay(int year, int daySeqOfYear);

7. int getDaySeqOfWeek(int year, int daySeqOfYear);

8. int getNextMonday(int year, int daySeqOfYear);

9. int getThisMonday(int year, int daySeqOfYear);

10. int getThisSunday(int year, int daySeqOfYear);

11. void printOneDay(int year, int daySeqOfYear);

## 实验步骤

1.编程思路如下(流程图)

图示

描述已自动生成

2.关键代码:

(1)得到每年1.1的星期数:

1. int getDaySeqOfJan1(int year)

2. {

3. return (year + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400) % 7;

// 公式,得到jan1星期

4. }

(2)计算最近的星期一和星期日:

1. int getThisMonday(int year ,int daySeqOfYear)

2. {

3. int i;

4. for (i = 0;i >= -6;i--) // 一个日子, +-6天 之内必然能碰到一周所有星期数

5. {

6. if (getDaySeqOfWeek(year,daySeqOfYear + i) == 1)

7. {

8. daySeqOfYear += i;

9. break;

10. }

11. }

12. return daySeqOfYear;

13. }

14. int getThisSunday(int year ,int daySeqOfYear)

15. {

16. int i;

17. for (i = 0;i <= 6;i++) // 一个日子,未来0-6天之内必然能碰到一周所有星期数

18. {

19. if (getDaySeqOfWeek(year,daySeqOfYear + i) == 7)

20. {

21. daySeqOfYear += i;

22. break;

23. }

24. }

25. return daySeqOfYear;

26.}

代码测试

### 测试点 测试整个代码的运行结果

1.测试步骤:

运行程序,多次输入特定值,检验程序运行状况

1. 特定值①: -1(负数)
2. 特定值②: 2024 4

2. 预期测试结果:

所有结果都准确无误

3. 实际测试结果:

文本

描述已自动生成

图片包含 文本

描述已自动生成

4. 测试结论:

日历系列(6)测试完成,能够实现所有的要求

## 实验结论

代码能达到功能目标

## 实验总结

代码能够完美实现所有要求

# 实验二(日历9)

## 实验任务

[实验目标]

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

[实验任务]

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

[要求函数]

改造main函数

## 实验步骤(流程图)

图示

中度可信度描述已自动生成

2.关键代码:

(1)初始化在年份数组范围内的每一天:

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

2. {

3. int year , yearLength ;

4. int i, j;

5.

6. for ( i = 0 ; i < YEAR\_NUM ; i ++ )

7. {

8. year = Years[i];

9. yearLength = isLeapYear( year ) ? 366 : 365;

10. for ( j = 0 ; j < yearLength; j ++ )

11. {

12. Days[i][j][0] = getMonth( year, j + 1 );

13. Days[i][j][1] = getDay( year, j + 1 );

14. Days[i][j][2] = getWeekSeqOfYear( year, Days[i][j][0], Days[i][j][1] );

15. Days[i][j][3] = getDaySeqOfWeek( year, j + 1 );

16. }

17. }

18. return;

19. }

20.

(2)打印生日所在周

1. void printOneWeek( int year, int month, int day, int Days[][366][4])

2. {

3. int daySeqOfYear = getDaySeqOfYear(year, month, day);

4. int weekSeq = getWeekSeqOfYear(year, month, day);

5. int daySeqOfWeek = getDaySeqOfWeek(year, daySeqOfYear);

6. int startDaySeqOfYear = daySeqOfYear - (daySeqOfWeek - 1);

7.

8. printf("\nBirthday in Year %d\n", year);

9. printf(" #M:%10s%10s%10s%10s%10s%10s%10s\n",

10. "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

11. printf("[%02d]", weekSeq);

12.

13. int i ;

14. for ( i = 0; i < 7; i++)

15. {

16. int currentDaySeqOfYear = startDaySeqOfYear + i;

17. if (currentDaySeqOfYear >= 1 && currentDaySeqOfYear <= (isLeapYear(year) ? 366 : 365))

18. {

19. printOneDay(year, currentDaySeqOfYear);

20. }

21. else

22. {

23. printf("%10s", " ");

24. }

25. }

26. printf("\n");

## 代码测试

### 测试点 输入多个日期测试代码效果

1.测试步骤:

运行程序,多次输入特定值,检验程序运行状况

1. 特定值①:

输入3 1

1. 特定值②:

输入12 31

2. 预期测试结果:

所有结果都准确无误

3. 实际测试结果:

日历

描述已自动生成

日历

描述已自动生成

4. 测试结论:

日历系列(9)测试完成,能够实现所有的要求

## 实验结论

代码能达到功能目标

## 实验总结

要注意特殊日期比如跨年和跨月,做出相应代码

# 实验三(日历10)

## 实验任务

[实验目标]

学习结构体的使用。将日期作为一种新的数据类型，对其进行“赋值”、“偏移天数”等特殊操作，体会“面向过程的编程”与“面向数据类型的编程”在设计思路上的差异。

[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。

[要求函数]

1. int isLeapYear(int year );

2. int getDaySeqOnJan1(int year);

3. int getDaysSeq(int year, int month, int day);

4. int getWeekSeqOfYear(int year, int month, int day);

5. int getMonth(int year, int daysSeqOfYear);

6. int getDay(int year, int daysSeqOfYear);

7. int getDaysSeqOfWeek(int year, int daysSeqOfYear);

8. void setYearArray(int yearArray[], int yearNum, int yearStart);

9. int isDay(int year, int month, int day);

10. Day setDay(int year, int month, int day);

11. Day getDayBefore( Day currentDay, int interval );

12. Day getDayAfter( Day currentDay, int interval );

13. int getTwoDaysInterval( Day startDay, Day endDay );

14. void printDay( Day currentDay, int displayFormat );

## 实验步骤

1.编程思路如下(流程图)

图示

描述已自动生成

2.关键代码:

(1)循环处理每个年份:

1. for ( year = YEAR\_MIN ; year < YEAR\_MIN + YEAR\_NUM ; year ++ )

2. {

3. int day = isDay(Years[year - YEAR\_MIN], inputMonth , inputDay);

4. if (day == -1)

5. {

6. printf("Invalid date.\n") ;

7. return 0;

8. }

9. // 打印标头

10. printf(" #:%12s%12s%12s%12s%12s%12s%12s", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

11. Day birthday = Years[year - YEAR\_MIN][day]; // 得到生日的所有信息

12. Day startday = getDayBefore(birthday, 2); // 得到开始日期

13. printDay(startday, DATE\_INFO\_FULL); // 打印

14. Day secondday = getDayBefore(birthday, 1); // 第二天

15. if (secondday.weekDay == 0)

16. {

17. printDay(secondday, DATE\_INFO\_FULL);

18. }

19. else

20. {

21. (printDay(secondday, DATE\_INFO\_BRIEF));

22. }

23. if (birthday.weekDay == 0)

24. {

25. printDay(birthday, DATE\_INFO\_FULL);

26. }

27. else

28. {

29. printDay(birthday, DATE\_INFO\_BRIEF);

30. }

31. printf("\n\n") ; // 增加美观性

32. }

(2)得到生日前的日期:

1. Day getDayBefore(Day currentDay, int interval)

2. {

3. Day newDay = currentDay;

4. newDay.daySeq -= interval;

5. if (newDay.daySeq < 0)

6. {

7. newDay.year--;

8. newDay.daySeq += isLeapYear(newDay.year) ? 366 : 365;

9. }

10. newDay.month = getMonth(newDay.year, newDay.daySeq);

11. newDay.day = getDay(newDay.year, newDay.daySeq);

12. newDay.weekSeq = getWeekSeq0fYear(newDay.year, newDay.month, newDay.day);

13. newDay.weekDay = getDaySeqOfWeek(newDay.year, newDay.daySeq);

14. return newDay;

15. }

(3)得到生日后的日期:

1. Day getDayAfter(Day currentDay, int interval)

2. {

3. Day newDay = currentDay;

4. newDay.daySeq += interval;

5. if (newDay.daySeq >= (isLeapYear(newDay.year) ? 366 : 365))

6. {

7. newDay.year++;

8. newDay.daySeq -= isLeapYear(newDay.year) ? 366 : 365;

9. }

10. newDay.month = getMonth(newDay.year, newDay.daySeq);

11. newDay.day = getDay(newDay.year, newDay.daySeq);

12. newDay.weekSeq = getWeekSeq0fYear(newDay.year, newDay.month, newDay.day);

13. newDay.weekDay = getDaySeqOfWeek(newDay.year, newDay.daySeq);

14. return newDay;

15. }

## 代码测试

### 测试点 循环处理每个年份的测试结果

1.测试步骤:

运行程序,多次输入特定值,检验程序运行状况

1. 特定值①:

输入3 2(3月2日)

1. 特定值②:

输入2 29(2月29日)

2. 预期测试结果:

所有结果都准确无误

3. 实际测试结果:

文本

描述已自动生成

文本

描述已自动生成

4. 测试结论:

日历系列(10)测试完成,能够实现所有的要求

## 实验结论

代码能达到功能目标

## 实验总结

实验中对于跨年和跨月等情况应该要仔细考虑,同时要注意多组函数的配合,使整体代码实现预期效果,最终代码效果显著,顺利完成所有要求

# 本课程学习总结

学习C语言的历程和总结

程序中常常出现无限死循环的情况，后来经过检查，常常是if语句和for循环嵌套的代码出错误。改进方法:一步一步慢慢写代码，边写边进行检测

代码规范：命名规范，同时要有各种间隔。

调试技巧：使用debug，并且，对于日历系列来说，可以把年份范围缩短为两年进行调试，这样更方便。

学习记录与心得:

C语言打开了我对编程世界认知的大门，学会了很多的控制语句以及各式各样的数据类型，我明白了用编程语言看世界的思想，同时，C语言的学习也为我今后各种编程语言的学习以及本专业的学习具有深远的影响，并且从C语言的学习过程中，我经历了各种各样的困难与挑战，但最终都成功的克服了，这些经历都是我宝贵的财富

# 附录

完整实验代码附在此处

1. 日历(6)实验

1. #include <stdio.h>

2. #include <stdlib.h>

3.

4. int getDaySeqOfYear( int year , int month , int day ); // 由年月日得到日期经历的总天数

5. int isLeapYear( int year ); // 看是不是闰年

6.

7. int getMonth( int year , int daySeqOfYear ); // 由天数得到月份

8. int getDay( int year , int daySeqOfYear ); // 由天数得到日期(日)

9.

10. int getDaySeqOfWeek( int year , int daySeqOfYear ); // 由天数得到星期

11. int getDaySeqOfJan1( int year ); // 计算每年1月1日是星期几

12.

13. int getThisMonday( int year , int daySeqOfYear ); //计算离dayseqofyear最近的周一(后)

14. int getThisSunday( int year , int daySeqOfYear ); //计算离dayseqofyear最近的周日(后)

15.

16. void printOneDay( int year , int daySeqOfYear ); // 打印日历主体部分

17.

18. int main( void )

19. {

20. int year;

21.

22. printf("Please int put the current year:\n");

23. scanf("%d",&year);// 输入年份

24.

25. if (year <= 0) // 负数年份在本项目中没有用,所以排除

26. {

27. printf("\nInput Error !!! The year should be positive \n\n");

28. return 0 ;

29. }

30.

31. printf("Please input the reference day jan.(1-31):\n");

32.

33. int referenceDay; // 得到推荐放假日,用来计算最近的周日

34. scanf("%d",&referenceDay);

35.

36. if (referenceDay < 1 || referenceDay > 31) // 防老六

37. {

38. printf("\n\nReferenceday input Error!!!\n");

39. return 0 ;

40. }

41.

42. printf("%20s"," "); // 打印校历标题

43. printf("Fall Semester Calendar of year %d\n",year);

44.

45. printf(" #W:"); // 打印表头

46. printf("%10s%10s%10s%10s%10s%10s%10s\n","Mon.","Tues.","Wed.","Thur.","Fri.","Sat.","Sun.");

47.

48. int startSeqOfYear; // 开学日

49. int endSeqOfYear; // 放学日

50.

51. startSeqOfYear = getThisMonday(year,getDaySeqOfYear(year,9,1)); // 开始日

52. endSeqOfYear = getThisSunday(year + 1,getDaySeqOfYear(year + 1,1,referenceDay)); // 结束日

53.

54. int daySeqOfYear;

55. int daySeqOfWeek;

56. int weekSeqOfSemester;

57.

58. for (daySeqOfYear = startSeqOfYear,daySeqOfWeek = 0,weekSeqOfSemester = 1;

59. ;daySeqOfYear++,daySeqOfWeek++,daySeqOfWeek %= 7 /\* 由此计算是周几 \*/)

60. {

61. if (daySeqOfWeek == 0)

62. {

63. printf("[%02d]",weekSeqOfSemester);

64. }

65.

66. printOneDay(year , daySeqOfYear);

67.

68. if (daySeqOfYear == 365 + isLeapYear(year))

69. // 跨年设置,年数加一,总天数归零(因为有daySeqOfYear++,所以用0)

70. {

71. year++;

72. daySeqOfYear = 0;

73. }

74.

75. if (daySeqOfWeek == 6)

76. {

77. printf("\n");

78. weekSeqOfSemester++;

79. }

80.

81. if (daySeqOfYear == endSeqOfYear)

82. {

83. break;

84. }

85. }

86.

87. return 0;

88. }

89.

90. int getDaySeqOfYear(int year,int month,int day)

91. {

92. int total;

93. total = day;

94. switch ( month )

95. {

96. case 12:

97. total += 30;

98. case 11:

99. total += 31;

100. case 10:

101. total += 30;

102. case 9:

103. total += 31;

104. case 8:

105. total += 31;

106. case 7:

107. total += 30;

108. case 6:

109. total += 31;

110. case 5:

111. total += 30;

112. case 4:

113. total += 31;

114. case 3:

115. total += isLeapYear(year) == 1 ? 29 : 28;

116. case 2:

117. total += 31;

118. case 1:

119. break;

120. }

121. // 也可以使用数组计算

122. return total;

123. }

124. int isLeapYear(int year)

125. {

126. if((year % 4 == 0 && year % 100 != 0)|| year % 400 == 0)

127. {

128. return 1;

129. }

130. else // 这么设置return值其实就是对应365 和 366

131. {

132. return 0;

133. }

134. }

135. int getMonth(int year,int daySeqOfYear)

136. {

137. int i;

138. int month;

139. for (i = 1; i <= 12;i++)

140. {

141. if ((daySeqOfYear >= getDaySeqOfYear(year,i,1) && daySeqOfYear < getDaySeqOfYear(year,i + 1,1))

142. || (daySeqOfYear >= getDaySeqOfYear(year,12,1) && daySeqOfYear <= 365 + isLeapYear(year)))

143. {// 上面if签一个括号是为了根据夹逼法得到月份,而后面一个括号专门为了12月

144. // (因为i= 12时,i+1 = 13,swich取不到,直接无效swich了)

145. month = i;

146. }

147. }

148. return month;

149. }

150. int getDay(int year,int daySeqOfYear)

151. {

152. int month;

153. month = getMonth(year,daySeqOfYear);

154. int day;

155. day = daySeqOfYear - getDaySeqOfYear(year,month,1) + 1; // 就这么算的,得到日期

156. return day;

157. }

158. int getDaySeqOfWeek(int year,int daySeqOfYear)

159. {

160. int Jan1;

161. Jan1 = getDaySeqOfJan1(year);

162.

163. return (daySeqOfYear + Jan1 - 2) % 7 + 1; // 就这么算的,得到星期

164.

165. }

166. int getDaySeqOfJan1(int year)

167. {

168. return (year + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400) % 7; // 公式,得到jan1星期

169. }

170. int getThisMonday(int year ,int daySeqOfYear)

171. {

172. int i;

173. for (i = 0;i >= -6;i--) // 一个日子, +-6天 之内必然能碰到一周所有星期数

174. {

175. if (getDaySeqOfWeek(year,daySeqOfYear + i) == 1)

176. {

177. daySeqOfYear += i;

178. break;

179. }

180. }

181. return daySeqOfYear;

182. }

183. int getThisSunday(int year ,int daySeqOfYear)

184. {

185. int i;

186. for (i = 0;i <= 6;i++) // 一个日子,未来0-6天之内必然能碰到一周所有星期数

187. {

188. if (getDaySeqOfWeek(year,daySeqOfYear + i) == 7)

189. {

190. daySeqOfYear += i;

191. break;

192. }

193. }

194. return daySeqOfYear;

195. }

196. void printOneDay(int year ,int daySeqOfYear)

197. {

198. int month;

199. int day;

200.

201. month = getMonth(year,daySeqOfYear);

202. day = getDay(year,daySeqOfYear);

203.

204. if (day == 1)

205. {

206. if (getDaySeqOfWeek(year,daySeqOfYear) == 6 || getDaySeqOfWeek(year,daySeqOfYear) ==7)

207. {

208. printf("%4s"," ");// 打印出01.01类似的效果

209. printf("%02d.%02d!",month,day);

210. }

211. else

212. {

213. printf("%5s"," ");

214. printf("%02d.%02d",month,day);

215. }

216. }

217.

218. else

219. {

220. if(getDaySeqOfWeek(year,daySeqOfYear) == 6 || getDaySeqOfWeek(year,daySeqOfYear) ==7)

221. {

222. printf("%9d!",day);

223. }

224. else

225. {

226. printf("%10d",day);

227. }

228. }

229. }

230.

1. 日历(9)实验

1. #include <stdio.h>

2. #include <stdlib.h>

3.

4. #define YEAR\_MIN 2020

5. #define YEAR\_MAX 2025

6. #define YEAR\_NUM YEAR\_MAX - YEAR\_MIN

7.

8. int getMonth( int year , int totalDay ) ;

9. void setYearArray ( int years[] , int yearNum , int yearMin ) ;

10. int getDaySeqOfYear( int year , int month , int day ) ;

11. int getMonthLength( int year , int month ) ;

12. int isLeapYear( int year ); // 看是不是闰年

13. int getDay(int year,int daySeqOfYear) ;

14. int getDaySeqOfWeek(int year,int daySeqOfYear) ;

15. int getDaySeqOfJan1( int year ); // 计算每年1月1日是星期几

16. int getWeekSeqOfYear( int year, int month , int day ) ;

17. void printOneDay(int year ,int daySeqOfYear) ;

18. void printOneWeek( int year, int month, int day, int Days[][366][4]) ;

19. void initialDays(int Years[], int Days[][366][4], int yearNum) ;

20.

21. int main( )

22. {

23. int inputMonth, inputDay, Years[YEAR\_NUM];

24.

25. int Days[YEAR\_NUM][366][4] = { 0 } ; // 初始化每年的每一天

26.

27. int totalNum = 0, weekendNum = 0;

28.

29. printf("Finding Birthday in year (%d-%d), please input the month and day:\n",

30. YEAR\_MIN, YEAR\_MAX);

31. scanf("%d%d", &inputMonth, &inputDay); // 得到生日

32.

33. if (inputMonth < 1 || inputMonth > 12 || inputDay < 1 || inputDay > 31)

34. {

35. printf("Sorry, the input month and day are invalid.\n");

36. return 0;

37. }

38. // 设置年份数组

39. setYearArray(Years, YEAR\_NUM, YEAR\_MIN);

40. // 初始化每年每日的信息

41. initialDays(Years, Days, YEAR\_NUM);

42.

43. int i ;

44. for ( i = 0; i < YEAR\_NUM; i++)

45. {

46. int year = Years[i];

47. // 打印本周

48. printOneWeek(year, inputMonth, inputDay, Days);

49.

50. int daySeqOfYear = getDaySeqOfYear(year, inputMonth, inputDay);

51. int daySeqOfWeek = getDaySeqOfWeek(year, daySeqOfYear);

52. if ( daySeqOfWeek == 6 || daySeqOfWeek == 7 )

53. {

54. weekendNum++;

55. }

56. totalNum++;

57. }

58.

59. printf("\nTotal %d birthdays are found, %d of them are in weekends.\n",

60. totalNum, weekendNum);

61. return 0;

62. }

63. // 设置年份数组

64. void setYearArray ( int years[] , int yearNum , int yearMin )

65. {

66. int i ;

67. for ( i = 0 ; i < yearNum ; i ++ )

68. {

69. years[i] = yearMin + i ;

70. }

71. }

72. // 初始化在年份数组范围内的每一天

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

74. {

75. int year , yearLength ;

76. int i, j;

77.

78. for ( i = 0 ; i < YEAR\_NUM ; i ++ )

79. {

80. year = Years[i];

81. yearLength = isLeapYear( year ) ? 366 : 365;

82. for ( j = 0 ; j < yearLength; j ++ )

83. {

84. Days[i][j][0] = getMonth( year, j + 1 );

85. Days[i][j][1] = getDay( year, j + 1 );

86. Days[i][j][2] = getWeekSeqOfYear( year, Days[i][j][0], Days[i][j][1] );

87. Days[i][j][3] = getDaySeqOfWeek( year, j + 1 );

88. }

89. }

90. return;

91. }

92. int getMonthLength( int year , int month )

93. {

94. // 二月要分类讨论

95. int dayLengthOfMonth2 = ( year % 4 == 0 && year % 400 != 0 ) ? 29 : 28 ;

96. int months[12] = {31,dayLengthOfMonth2,31,30,31,30,31,31,30,31,30,31} ;

97. return months[month - 1] ;

98. }

99. int getDaySeqOfYear( int year , int month , int day )

100. {

101. int i ;

102. int total = day ;

103. for ( i = 1 ; i < month ; i ++ )

104. {

105. total += getMonthLength( year , i ) ;

106. }

107. return total ;

108. }

109. int getMonth(int year,int daySeqOfYear)

110. {

111. int i ;

112. int month ;

113. for (i = 1 ; i <= 12 ; i++)

114. {

115. if ((daySeqOfYear >= getDaySeqOfYear(year,i,1) && daySeqOfYear < getDaySeqOfYear(year,i + 1,1))

116. || (daySeqOfYear >= getDaySeqOfYear(year,12,1) && daySeqOfYear <= 365 + isLeapYear(year)))

117. {// 上面if签一个括号是为了根据夹逼法得到月份,而后面一个括号专门为了12月

118. // (因为i= 12时,i+1 = 13,swich取不到,直接无效switch了)

119. month = i;

120. }

121. }

122. return month;

123. }

124. int isLeapYear(int year)

125. {

126. if((year % 4 == 0 && year % 100 != 0)|| year % 400 == 0)

127. {

128. return 1;

129. }

130. else // 这么设置return值其实就是对应365 和 366

131. {

132. return 0;

133. }

134. }

135. int getDay(int year,int daySeqOfYear)

136. {

137. int month;

138. month = getMonth(year,daySeqOfYear);

139. int day;

140. day = daySeqOfYear - getDaySeqOfYear(year,month,1) + 1; // 就这么算的,得到日期

141. return day;

142. }

143. int getDaySeqOfWeek(int year,int daySeqOfYear)

144. {

145. int Jan1;

146. Jan1 = getDaySeqOfJan1(year);

147.

148. return (daySeqOfYear + Jan1 - 2) % 7 + 1 ; // 就这么算的,得到星期

149.

150. }

151. int getDaySeqOfJan1(int year)

152. {

153. return (year + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400) % 7; // 公式,得到jan1星期

154. }

155. int getWeekSeqOfYear( int year, int month , int day )

156. {

157. int weekdayOfJan1 = getDaySeqOfJan1( year ) ; // 获取1月1日是星期几

158. int daysInFirstWeek = 8 - weekdayOfJan1 ; // 第一周的天数

159.

160. if ( getDaySeqOfYear( year , month , day ) <= daysInFirstWeek)

161. {

162. // 第n天在第一周

163. return 1;

164. }

165.

166. else

167. {

168. // 第n天在第一周之后，计算周次

169. int remainingDays = getDaySeqOfYear( year , month , day ) - daysInFirstWeek;

170. int fullWeeks = remainingDays / 7;

171. int additionalDays = remainingDays % 7;

172.

173. if (additionalDays > 0)

174. {

175. // 如果还有额外的天数，周次加1

176. return fullWeeks + 2; // 加2是因为第一周已经算了一个周次

177. }

178. else

179. {

180. // 没有额外的天数，直接返回完整的周次数加1（第一周）

181. return fullWeeks + 1;

182. }

183. }

184. }

185. void printOneDay(int year ,int daySeqOfYear)

186. {

187. int month;

188. int day;

189.

190. month = getMonth(year,daySeqOfYear);

191. day = getDay(year,daySeqOfYear);

192.

193. if (day == 1)

194. {

195. if (getDaySeqOfWeek(year,daySeqOfYear) == 6 || getDaySeqOfWeek(year,daySeqOfYear) ==7)

196. {

197. printf("%4s"," ");// 打印出01.01类似的效果

198. printf("%02d.%02d!",month,day);

199. }

200. else

201. {

202. printf("%5s"," ");

203. printf("%02d.%02d",month,day);

204. }

205. }

206.

207. else

208. {

209. if(getDaySeqOfWeek(year,daySeqOfYear) == 6 || getDaySeqOfWeek(year,daySeqOfYear) ==7)

210. {

211. printf("%9d!",day);

212. }

213. else

214. {

215. printf("%10d",day);

216. }

217. }

218. }

219. void printOneWeek( int year, int month, int day, int Days[][366][4])

220. {

221. int daySeqOfYear = getDaySeqOfYear(year, month, day);

222. int weekSeq = getWeekSeqOfYear(year, month, day);

223. int daySeqOfWeek = getDaySeqOfWeek(year, daySeqOfYear);

224. int startDaySeqOfYear = daySeqOfYear - (daySeqOfWeek - 1);

225.

226. printf("\nBirthday in Year %d\n", year);

227. printf(" #M:%10s%10s%10s%10s%10s%10s%10s\n",

228. "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

229. printf("[%02d]", weekSeq);

230.

231. int i ;

232. for ( i = 0; i < 7; i++)

233. {

234. int currentDaySeqOfYear = startDaySeqOfYear + i;

235. if (currentDaySeqOfYear >= 1 && currentDaySeqOfYear <= (isLeapYear(year) ? 366 : 365))

236. {

237. printOneDay(year, currentDaySeqOfYear);

238. }

239. else

240. {

241. printf("%10s", " ");

242. }

243. }

244. printf("\n");

245. }

246.

三、日历(10)实验

1. #include <stdio.h>

2. #define YEAR\_NUM 5 // 年份范围

3. #define YEAR\_MIN 2021 // 最小年份

4.

5. // 定义一个日期结构体

6. typedef struct

7. {

8. int year; // 年份

9. int daySeq; // 得到截至该日期的总天数

10. int month; // 月份

11. int day; // 日期

12. int weekSeq; // 周数

13. int weekDay; // 星期数

14. } Day;

15.

16. // 枚举

17. typedef enum

18. {

19. DATE\_INFO\_FULL,

20. DATE\_INFO\_BRIEF,

21. } DateFormat;

22.

23. int getDaySeqOnJan1(int year); // 得到选定年度的1.1日期对应的星期数

24.

25. int isLeapYear(int year); // 判断是不是闰年

26.

27. int getMonthLength(int year, int month); //得到月份的日长度

28. int getDaySeq(int year, int month, int day); // 得到截至该日期的总天数

29. int getWeekSeq0fYear(int year, int month, int day); // 得到该日期对应的星期数

30.

31. int getMonth(int year, int daySeqOfYear); // 得到月份

32. int getDay(int year, int daySeqOfYear); // 得到日期

33. int getDaySeqOfWeek(int year, int daySeq0fYear); // 得到星期数

34.

35. // 初始化年数组

36. void setYearArray(Day yearArray[][366], int yearNum, int yearStart);

37.

38. // 判断是否存在这一天

39. int isDay(Day year[], int month, int day);

40.

41. Day getDayBefore(Day currentDay, int interval);

42. Day getDayAfter(Day currentDay, int interval);

43.

44. // 打印展示

45. void printDay(Day currentDay, int displayFormat);

46.

47. int main( void )

48. {

49. Day Years[YEAR\_NUM][366] = { 0 }; // 假设的年份数组

50. int inputMonth, inputDay ;

51. setYearArray(Years , YEAR\_NUM , YEAR\_MIN ) ;

52.

53. // 用户输入生日的月份和日期

54. printf("Enter birth month: ");

55. scanf("%d", &inputMonth);

56.

57. printf("Enter birth day: ");

58. scanf("%d", &inputDay);

59.

60. int year ;

61. for ( year = YEAR\_MIN ; year < YEAR\_MIN + YEAR\_NUM ; year ++ )

62. {

63. int day = isDay(Years[year - YEAR\_MIN], inputMonth , inputDay);

64. if (day == -1)

65. {

66. printf("Invalid date.\n") ;

67. continue ;

68. }

69. // 打印标头

70. printf(" #:%12s%12s%12s%12s%12s%12s%12s", "Mon.", "Tues.", "Wed.", "Thur.", "Fri.", "Sat.", "Sun.");

71. Day birthday = Years[year - YEAR\_MIN][day]; // 得到生日的所有信息

72. Day startday = getDayBefore(birthday, 2); // 得到开始日期

73. printDay(startday, DATE\_INFO\_FULL); // 打印

74. Day secondday = getDayBefore(birthday, 1); // 第二天

75. if (secondday.weekDay == 0)

76. {

77. printDay(secondday, DATE\_INFO\_FULL);

78. }

79. else

80. {

81. (printDay(secondday, DATE\_INFO\_BRIEF));

82. }

83. if (birthday.weekDay == 0)

84. {

85. printDay(birthday, DATE\_INFO\_FULL);

86. }

87. else

88. {

89. printDay(birthday, DATE\_INFO\_BRIEF);

90. }

91. printf("\n\n") ; // 增加美观性

92. }

93. return 0;

94. }

95.

96. int getDaySeq(int year, int month, int day)

97. {

98. int totalDays = 0;

99. int i ;

100. for ( i = 1; i < month; i++)

101. {

102. totalDays += getMonthLength(year, i);

103. }

104. totalDays += day;

105. return totalDays;

106. }

107. int getMonthLength(int year, int month)

108. {

109. int daysInMonth[12] = {31, 28 + isLeapYear(year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

110. return daysInMonth[month - 1];

111. }

112. int getWeekSeq0fYear(int year, int month, int day)

113. {

114. return (getDaySeqOnJan1(year) + getDaySeq(year, month, day) - 1) / 7 + 1;

115. }

116. int isLeapYear(int year)

117. {

118. if( ( year % 4 == 0 && year % 100 != 0) || (year % 400 == 0) )

119. {

120. return 1 ;

121. }

122. return 0 ;

123. }

124. int getDaySeqOnJan1(int year)

125. {

126. int totalDays = 0;

127. int i ;

128. for ( i = 1900; i < year; i++)

129. {

130. totalDays += isLeapYear(i) ? 366 : 365;

131. }

132. return totalDays % 7; // 0 对应周一 , 6 对应周日

133. }

134. int getMonth(int year, int daySeq0fYear)

135. {

136. int month = 1;

137. while (daySeq0fYear > getMonthLength(year, month))

138. {

139. daySeq0fYear -= getMonthLength(year, month);

140. month++;

141. }

142. return month;

143. }

144. int getDay(int year, int daySeq0fYear)

145. {

146. int month = getMonth(year, daySeq0fYear);

147. return daySeq0fYear - getDaySeq(year, month, 1) + 1;

148. }

149. int getDaySeqOfWeek(int year, int daySeq0fYear)

150. {

151. return (getDaySeqOnJan1(year) + daySeq0fYear - 1) % 7;

152. }

153. void setYearArray(Day yearArray[][366], int yearNum, int yearStart)

154. {

155. int i ;

156. for (i = 0; i < yearNum; i++)

157. {

158. int j ;

159. for (j = 0; j < 366; j++)

160. {

161. yearArray[i][j].year = yearStart + i;

162. yearArray[i][j].daySeq = j;

163. yearArray[i][j].month = getMonth(yearArray[i][j].year, yearArray[i][j].daySeq);

164. yearArray[i][j].day = getDay(yearArray[i][j].year, yearArray[i][j].daySeq);

165. yearArray[i][j].weekSeq = getWeekSeq0fYear(yearArray[i][j].year, yearArray[i][j].month, yearArray[i][j].day);

166. yearArray[i][j].weekDay = getDaySeqOfWeek(yearArray[i][j].year, yearArray[i][j].daySeq);

167. }

168. }

169. }

170. int isDay(Day year[], int month, int day)

171. {

172. int i ;

173. for ( i = 0; i < 366; i++) {

174. if (year[i].month == month && year[i].day == day) {

175. return i;

176. }

177. }

178. return -1;

179. }

180. Day getDayBefore(Day currentDay, int interval)

181. {

182. Day newDay = currentDay;

183. newDay.daySeq -= interval;

184. if (newDay.daySeq < 0)

185. {

186. newDay.year--;

187. newDay.daySeq += isLeapYear(newDay.year) ? 366 : 365;

188. }

189. newDay.month = getMonth(newDay.year, newDay.daySeq);

190. newDay.day = getDay(newDay.year, newDay.daySeq);

191. newDay.weekSeq = getWeekSeq0fYear(newDay.year, newDay.month, newDay.day);

192. newDay.weekDay = getDaySeqOfWeek(newDay.year, newDay.daySeq);

193. return newDay;

194. }

195. Day getDayAfter(Day currentDay, int interval)

196. {

197. Day newDay = currentDay;

198. newDay.daySeq += interval;

199. if (newDay.daySeq >= (isLeapYear(newDay.year) ? 366 : 365))

200. {

201. newDay.year++;

202. newDay.daySeq -= isLeapYear(newDay.year) ? 366 : 365;

203. }

204. newDay.month = getMonth(newDay.year, newDay.daySeq);

205. newDay.day = getDay(newDay.year, newDay.daySeq);

206. newDay.weekSeq = getWeekSeq0fYear(newDay.year, newDay.month, newDay.day);

207. newDay.weekDay = getDaySeqOfWeek(newDay.year, newDay.daySeq);

208. return newDay;

209. }

210. void printDay(Day currentDay, int displayFormat)

211. {

212. if (displayFormat == DATE\_INFO\_FULL)

213. {

214. printf("\n#%02d:", currentDay.weekSeq);

215. int i ;

216. for ( i = 0; i < currentDay.weekDay; i++)

217. {

218. printf("%12s", " ");

219. }

220. printf("%2s%04d.%02d.%02d", " ", currentDay.year , currentDay.month, currentDay.day);

221. }

222. else if (displayFormat == DATE\_INFO\_BRIEF)

223. {

224. printf("%2s%04d.%02d.%02d", " ", currentDay.year , currentDay.month, currentDay.day);

225. }

226. }