

【注意:】

- 1、本次作业**不允许**使用后续课程的知识点,包括但不限于**全局变量**、数组、结构体、类等相关概念!!!
- 2、除明确要求外,已学过的知识中,不允许使用 goto
- 3、cstdio 及 cmath 中的系统函数**可以**直接使用,包括课上未介绍过的,具体可自行查阅相关资料
- 4、除明确要求外,所有 cpp 源程序不允许使用 scanf/printf 进行输入/输出
- 5、多编译器下均要做到“0 errors, 0 warnings”
- 6、部分题目要求 C 和 C++两种方式实现,具体见网页要求
- 7、输出为浮点数且未指定格式的,均要求为 double 型,C++为 cout 缺省输出,C 为 %lf 的缺省输出
- 8、认真阅读格式要求及扣分说明!!!

【输出格式要求:】

- 1、为方便机器自动判断正确性,作业有一定的输入输出格式要求(但不同于竞赛的无任何提示)
- 2、每个题目见具体说明,必须按要求输入和输出,不允许有偏差
- 3、没有特别说明的情况下,最后一行有效输出的最后有一个 endl
- 4、本次作业的**比对要求**为 txt_compare 在 --trim right 下与 demo 做到**完全一致**

【重要提醒:】

本周有一个学习使用 VisualStudio2022 下 Debug 工具的长期性作业,提交时间是第 17 周,但建议先看一下其中的视频,学会 Debug 的基本使用,对后续作业有**极大帮助!!!**

补充:

- 1、题目同 3-b6 (人民币转大写),要求 0-9 对应的大写数字只能用函数 daxie() 输出,除此函数外,**不允许**任何地方输出“零-玖”,其它内容(拾佰仟万圆角分整)可自行组织输出

【要求:】1、4-b1.cpp 的部分程序已给出

2、提供 4-b1-demo.exe 供参考(cmd 下运行,如有理解歧义,以 demo 为准)

3、不考虑输入错误,输入输出格式要求同 3-b6,保证 3-b6 中的所有测试数据均通过

- 2、用蔡勒(Zeller)公式将公元纪元的年月日转为星期

蔡勒(Zeller)公式可将年月日转为星期,0-6 分别表示星期日-星期六

公式为: $w=y+[y/4]+[c/4]-2c+[13(m+1)/5]+d-1$

公式中的符号含义如下,

w: 星期;

c: 世纪-1(即 4 位年份的前 2 位[注: 1900-1999 称为 20 世纪]);

y: 年(即 4 位年份的后 2 位);

m: 月(m 大于等于 3, 小于等于 14, 即在蔡勒公式中, 某年的 1、2 月要看作上一年的 13、14 月来计算, 比如 2003 年 1 月 1 日要看作 2002 年的 13 月 1 日来计算);

d: 日;

[]代表取整, 即只要整数部分。

(C 是世纪数-1, y 是年份后两位, M 是月份, d 是日数。1 月和 2 月要按上一年的 13 月和 14 月来算, 这时 C 和 y 均按上一年取值。)

算出来的 w (w 若为负数则加 7 的倍数后转为正数) 除以 7, 余数是几就是星期几。如果余数是 0, 则为星期日。

例 1：以 2014 年 4 月 2 日为例，用蔡勒（Zeller）公式进行计算，过程如下：

$$\begin{aligned}w &= y + [y/4] + [c/4] - 2c + [26(m+1)/10] + d - 1 \\&= 14 + [14/4] + [20/4] - 2 \times 20 + [13 \times (4+1)/5] + 2 - 1 \\&= 14 + [3.5] + 5 - 40 + [13] + 2 - 1 \\&= 14 + 3 + 5 - 40 + 13 + 2 - 1 \\&= -4 \\&= (-4 + 7) \% 7 \\&= 3 \quad (\text{星期三})\end{aligned}$$

【注：】罗马教皇格里高利十三世在 1582 年组织了一批天文学家，根据哥白尼日心说计算出来的数据，对儒略历作了修改。将 1582 年 10 月 5 日到 14 日之间的 10 天宣布撤销，继 10 月 4 日之后为 10 月 15 日。后来人们将这一新的历法称为“格里高利历”，也就是今天世界上所通用的历法，简称格里历或公历。因此蔡勒（Zeller）公式适用于 1582 年 10 月 15 日之后

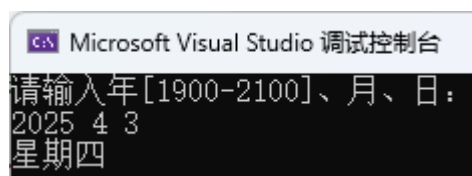
要求转换过程由函数 `int zeller(int y, int m, int d)` 完成，main 函数只负责输入年月日、检查年月日是否合法（假设年份范围 1900-2100）以及输出转换后的星期值。

输入/输出格式要求：多行（正确输入/错误输入不同）

Line1：输入提示“请输入年[1900-2100]、月、日：”

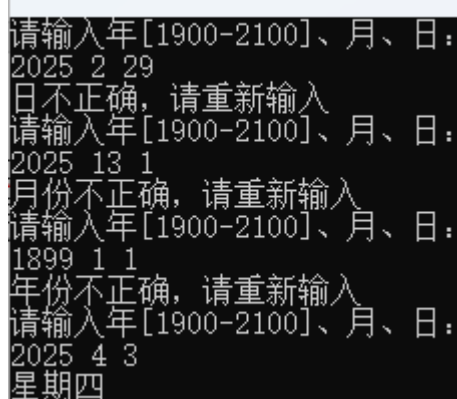
Line2：键盘输入的年月日（三个值一次性输入）

Line3：求得的星期值（中文）



```
Microsoft Visual Studio 调试控制台
请输入年[1900-2100]、月、日：
2025 4 3
星期四
```

Microsoft Visual Studio 调试控制台



```
Microsoft Visual Studio 调试控制台
请输入年[1900-2100]、月、日：
2025 2 29
日不正确，请重新输入
请输入年[1900-2100]、月、日：
2025 13 1
月份不正确，请重新输入
请输入年[1900-2100]、月、日：
1899 1 1
年份不正确，请重新输入
请输入年[1900-2100]、月、日：
2025 4 3
星期四
```

注意：屏幕显示与重定向的结果是不一样的，输出重定向文件不含输入信息（下同）

参考测试数据如下：

1900. 1. 1	一	1900. 2. 28	三
1900. 2. 29	非法	1900. 12. 31	一
2000. 1. 15	六	2000. 2. 28	一
2000. 2. 29	二	2000. 4. 13	四
2012. 2. 29	三	2013. 7. 12	五
2014. 2. 28	五	2014. 2. 29	非法
2014. 3. 6	四	2014. 4. 7	一
2018. 11. 1	四	2099. 12. 31	四

- 【要求：】
- 1、任意输入错误(非法、不合理值)均给出一行错误提示并重新输入年月日三个值，非法及不合理的处理逻辑参考第 3 章的作业
 - 2、年月日的错误按顺序依次判断
 - 3、给出 4-b2-demo.exe 供参考（cmd 下运行，如有理解歧义，以 demo 为准）
 - 4、demo 输入年月日时仍然用空格分开，不是小数点(例：1900 1 1)

3、从键盘输入年份和月份，打印该月的月历

输入/输出格式要求：多行（正确输入/错误输入不同）
Line1: 输入提示“请输入年[1900-2100]、月”
Line2: 键盘输入的年月（两个值一次性输入）
Line3: 空行
Line4 ~ : 打印的月历

【要求：】1、年份限定在 1900-2100 之间

- 2、用函数 zeller（必须与 4-b2 的同名函数实现完全一致）求某月 1 日的星期，用函数 calendar 打印月历，主函数负责输入年月的值、检查合理性以及调用函数
- 3、任意输入错误（非法、不合理值）均给出一行错误提示并重新输入年月两个值，非法及不合理的处理逻辑参考第 3 章的作业
- 4、年月错误按顺序依次判断
- 5、给出 4-b3-demo.exe 供参考（多个，均为 cmd 下运行，有理解歧义则以 demo 为准）
- 6、给出 4-b4.cpp 的部分代码（格式部分），可直接复制使用，不算打表
- 7、demo 输入年月时仍然用空格分开，不是小数点（例：1900 1）
- 8、注：不要忘了测试 2023 9（最后一天是周六）

4、写一个用键盘按键控制光标移动的程序，要保证光标的移动范围不超过边框的位置

【要求：】1、4-b4.cpp/4-b4.c 的部分源程序已给出

- 2、给出可执行文件（4-b4-demo.exe，cmd 下运行）供参考，要求达到以下要求
- 2.1、通过一个简易菜单进行选择，完成 demo 的 0-4 项，形式如下所示

```
1. 用I、J、K、L键控制上下左右（大小写均可，按左箭头光标不允许下移，边界停止）
2. 用I、J、K、L键控制上下左右（大小写均可，按左箭头光标不允许下移，边界回绕）
3. 用箭头键控制上下左右（按大写HPKM不允许移动光标，边界停止）
4. 用箭头键控制上下左右（按大写HPKM不允许移动光标，边界回绕）
0. 退出
[请选择0-4] _
```

- 2.2、菜单项选择时，不需要按回车键
 - 2.3、程序开始时光标停在由“*”组成的边框的中心位置
 - 2.4、用字母 I/J/K/L（**大小写均可**）或上下左右箭头键进行光标的移动
 - 2.5、若移动到有字母的位置，按空格键可以消除该字母（消除后光标位置不变）
 - 2.6、按 Q（大小写均可）退出，在下方“*”后第 5 行打印“游戏结束，按**回车**键返回菜单。”后，等待按**回车**键返回（要求：按其他键则无任何反应，也不退出）
 - 2.7、第 2/4 项要求光标环绕，注意光标不要停到“*”组成的边框上
- 3、本题不通过输入输出重定向方式自动判卷，允许菜单的文字/符号与 demo 有差异
 - 4、4-b4-demo.exe 菜单 5-8 项的“允许按左箭头时向下移动”/“大写 HPKM”可移动，不是让你**刻意做出**这种错误，而是如果你遵守了作业的要求去读键，不做任何特殊处理，这个问题就**必然出现**!!!（因此，菜单 1-4 就是要求进行特殊处理来解决问题）
 - 5、**按键时注意不要处于中文输入法状态，否则可能出现光标消失的情况，如果碰到这种情况，关闭 exe 后再次运行**

【提示：】

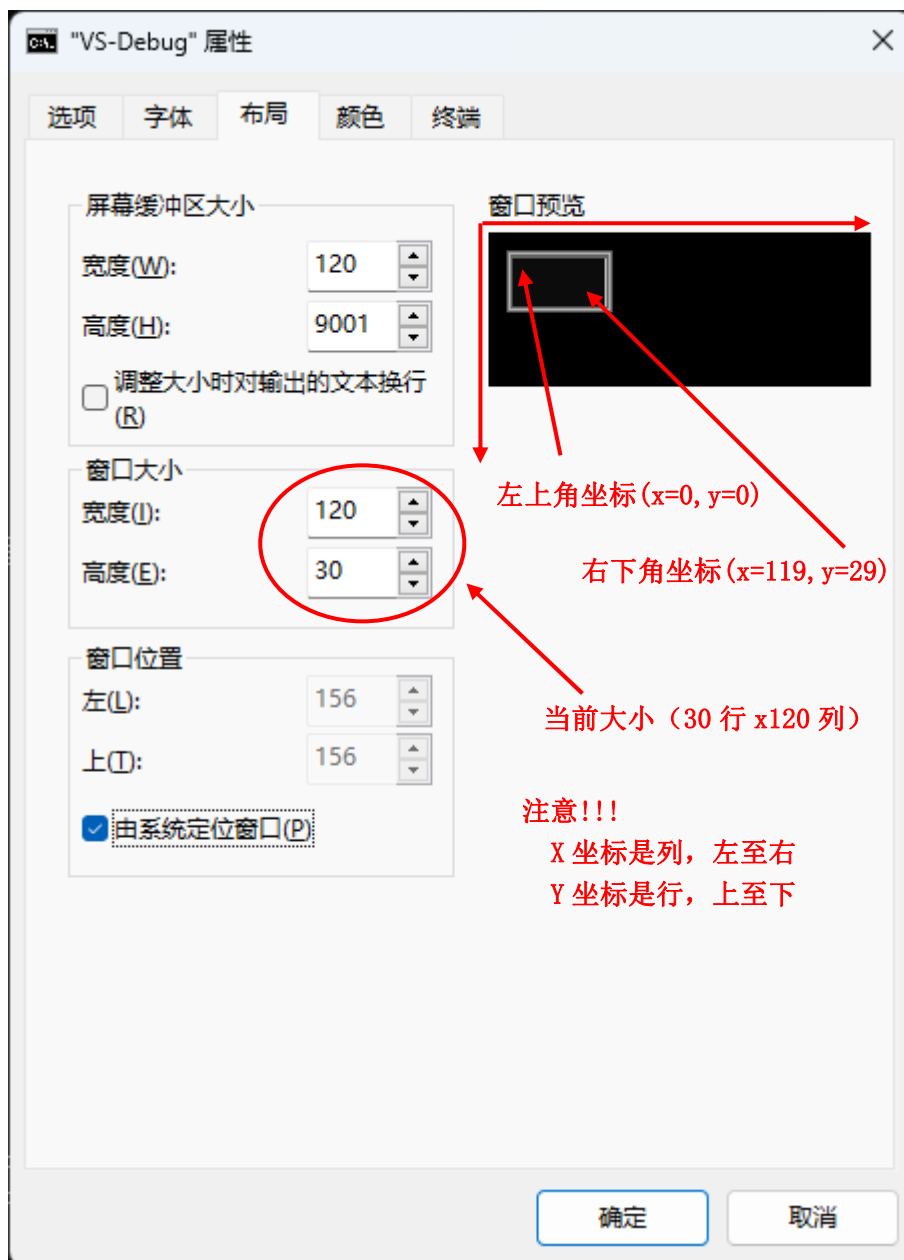
- 1、读键盘**直接用**_getch()函数即可，**不需要**自己去网上找其它函数
- 2、该函数读键盘时，所有按键都只返回一个键码吗？如何验证？
- 3、比较下 VS/Dev 下该函数的返回值完全相同吗？
- 4、将_getch()的返回值赋值给 char/unsigned char/int，为什么会有不同？

【函数分解要求:】

- 1、源码中已经给出的 cls/gotoxy/showch/init_border 函数不准修改，也不允许自己再写相同/相似功能函数
- 2、菜单显示及选择做成一个函数 menu()，返回选择的 0-4 项
- 3、用 i/j/k/l 移动的两项共用一个函数 move_by_ijkl()，光标是否回绕通过参数选择
- 4、用箭头键移动的两项共用一个函数 move_by_arrow，光标是否回绕通过参数选择
- 5、**鼓励**将 3/4 合并为一个函数，即光标的移动用一个函数完成，通过参数选择来决定用什么键移动光标、是否回绕（**本项不做强制要求**）
- 6、主函数负责循环调用菜单函数、根据菜单函数的返回值决定调用哪个光标移动函数或退出

【cmd 窗口的坐标说明:】

在屏幕保持初始大小（未在 cmd 窗口左上角的属性菜单设置过）的情况下，屏幕坐标为左上角 (0, 0)，右下角 (119, 29)，具体可见下面的图示，注意是整个窗口，不是“*”号组成的边框



【思考:】

- 1、本题给出的部分程序中，有在“*”组成的框中随机生成 20 个字母的代码段，每次运行完成后，一定能在框中生成 20 个字母吗？有没有可能不到 20 个字母？为什么？（不需要提交答案，想明白为什么即可）

【伪随机数的基本概念:】

- 1、srand 函数称为种子函数，每调用一次则生成一个种子
- 2、rand 函数的功能是产生一个伪随机数，每调用一次，VS 下会得到一个 0-32767 间的 int 型数据（其它编译器可能不是这个数据范围）
- 3、伪随机数：不是真正的随机数，“伪”=种子相同则后续随机数序列相同，因此称为伪随机数
- 4、阅读并运行“4-b4-伪随机数使用示例.cpp”中关于伪随机数产生的程序段，并结合下面的示例程序来理解

<pre>//本例用当前系统时间做种子，每次运行，随机序列不相同 #include <iostream> #include <time.h> #include <windows.h> using namespace std; int main() { /* 生成伪随机数的种子，程序开始时执行一次即可 */ srand((unsigned int)(time(0))); /*生成 10 个伪随机数 */ for (int i=0; i<10; i++) cout << rand() << endl; return 0; }</pre>	<pre>//本例用固定值做种子，每次运行，随机序列相同 #include <iostream> #include <time.h> #include <windows.h> using namespace std; int main() { /* 生成伪随机数的种子，程序开始时执行一次即可 */ srand(111); //可以改为任意 unsigned int 整数 /*生成 10 个伪随机数 */ for (int i=0; i<10; i++) cout << rand() << endl; return 0; }</pre>
--	---

结论（不需要交，但要弄懂，后面会用）:

- 1、当 srand 的种子不变时，产生的伪随机数序列____（变/不变）
- 2、如果程序需要产生大量随机数，则 srand_____（仅需要在最开始调用一次/每次 rand 之前都需要调用）

【编译器要求:】

		编译器VS	编译器Dev
4-b1.cpp	人民币转大写	Y	Y
4-b2-1.c	蔡勒公式(C方式)	Y	Y
4-b2-2.cpp	蔡勒公式	Y	Y
4-b3.cpp	打印月历	Y	Y
4-b4-1.c	键盘控制光标移动(C方式)	Y	Y
4-b4-2.cpp	键盘控制光标移动	Y	Y

【作业要求:】

- 1、4月9日前网上提交本次作业
- 2、每题所占平时成绩的具体分值见网页
- 3、超过截止时间提交作业会自动扣除相应的分数，具体见网页上的说明