Date类说明

1. 时间格式为“yyyy-mm-dd/hh:mm”。
2. **默认构造函数构造一个年月日等都是0的Date类对象**
3. **构造函数及 `stringToDate` 无需检查逻辑合法性（如闰年问题等）**
4. isValid用于判断时间是否合理，如年月日小时分的合理范围（**年的范围是 1000~9999**）以及闰年问题等，但不比较输入的时间是否过时。
5. stringToDate中，类似于 2016-7-08/01:00 的输入是不符合格式的。如果输入的字符串不符合格式，就**构造一个年月日等都是0的Date 类**（相当于调用默认构造函数）。
6. dateToString中，**转换前需要检查日期是否合法，若不合法，直接返回“0000-00-00/00:00”**。
7. 比较操作中无需考虑日期不合法问题（不会针对此进行测试）。
8. **以上提及的“合法”特指”逻辑上的合法”，与格式无关**

Storage 类说明

1. Storage 类中 writeToFile 和 readFromFile 操作的存储数据的两个文件的数据格式有什么要求呢？
   数据文件保存在工程目录的“data”文件夹下；另外，规定两个文件的数据格式要求如下：
   a) 每个项都必须加双引号；
   b) 一条完整的记录中间不会出现换行符，记录末尾有一个换行符，即一行一条记录。最后一个记录末尾有无换行符均可（不会对此进行测试）
   c) 输入的项的值不会包含双引号, “&”字符和逗号；
   d) 数据文件列的顺序是固定的。
   e) user.csv 文件中每条记录的格式是（引号为英文引号，**相邻列间无空格分隔**）：
   “username”,”password”,”email”,”phone”
   一个用户名为 name,密码为 123，邮箱是 123@123.com,手机是 123456 的记录在数据文件中的记录如下
   “name”,“123”,“123@123.com”,“123456”
   f) meeting.csv 文件中每条记录的格式是（引号为英文引号，**相邻列间无空格分隔，若有多个参与者，参与者之间使用“&”字符分隔**）：
   ”sponsor”,”participators”,”startDate”,”endDate”,”title”
   一个发起者为 A,参与者有 B 和 C，邮箱是开始时间是 2016-08-01/00:00，结束时间是 2016-08-01/12:00，标题为 meeting 的记录在数据文件中的记录如下：
   “A”,“B&C”,“2016-08-01/00:00”,“2016-08-01/12:00”,”meeting“
   g) **写入文件的格式中不包含数据项的名称**
2. 下面这个函数接口的 filter 和 switcher 是什么？
   int updateUser(std::function<bool(const User&)> filter, std::function<void(User&)> switcher);
   回答：
   std::function是c++11 里引入的，编译的时候需要加上参数-std=c++11。
   从接口可以看出，filter 和 switcher 也就是函数 updateUser 的参数，
   也是 function 模板类的一个实例；

就 filter 来说，function<bool(const User&)>指的是：filter 是一个返回值为 bool、参数为 const User&的函数；而 switcher 就是：返回值为 void、参数为 const User&的函数；至于这两个函数到底实现什么样的功能是由调用 updateUser 这个函数时传入的参数决定的；
可以把 filter 看成是一个过滤函数，当 filter 返回 true时就对这个 User 进行 switcher 函数的操作；
在其他函数接口里面的 filter 也和上面讲的 filter 相似。

3. sync 这个接口是用来干什么的？
   sync意思是同步，具体就是立即把保存在内存里的数据写入到文件，
   防止文件数据的不一致（为什么？所有访问数据不都是通过 Storage 来访问的吗？怎么会不一致？答：那些文件是存储在电脑磁盘上的，其他程序也是可以打开那个文件来进行读写操作，为了防止其他程序读这个文件时数据的不一致，调用这个函数立即对文件进行写入来进行文件同步）
   当m_dirty为真（数据被修改），调用writeToFile并置m_dirty为假即可。
4. 本地测试时/tmp/meetings.csv读写失败？
   机器测试时路径需为Path::meetingPath，而提供头文件中Path::meetingPath为”/tmp/meetings.csv”是评测需要，本地测试时修改为”data/meetings.csv”即可，user.csv同理。
5. createMeeting等中是否需考虑需求文档中提及的各种限制？
   数据层不考虑业务逻辑层的事情，请学习“三层架构”。

AgendaService类说明

1. 用户注册时，mail 和 phone 不需要判断唯一性和格式的正确性
2. **会议的title需确保全局唯一**
3. 会议的发起者和参与者需为**已注册用户**
4. 不允许创建会议为一个时间点，比如8:00—8:00
5. 关于找出一个时间段（**参数允许退化为一个时间点**）内的会议，**只要有时间点重叠，都要算上去。**如查询区间为2016-07-08/12:00至2016-07-08/12:00时，时间为2016-07-08/11:00至2016-07-08/12:00的会议可以被查询到。
6. 若查询过程中输入的日期（格式或逻辑）不合法，直接返回空列表
7. 创建会议，在判断会议是否重叠时，**允许会议开始时间是另一个会议的结束时间**，即一个会议结束后马上开始另一个会议。可以创建12:00-13:00、13:00-15:00 这样的时间段有共同端点的会议
8. **取消会议为deleteMeeting，只能由发起者操作；退出会议为quitMeeting，只能由参与者操作**
9. meetingQuery查询用户作为**发起者或参与者**的会议
10. deleteAllMeetings时，若删除会议数量为0，返回true或false均可（不作测试）

一些提醒

1. 遇到RE错误的同学可以检查一下有没有“迭代器失效”的问题，例如一边遍历容器一边erase（这种场景建议使用 `remove_if` ）。另外，建议**自己编写main函数，并用valgrind、gdb进行调试。**
2. 提交评测前务必在本地**确保编译通过**（自己编写一个main函数测试），以免白白浪费评测机会。

3. 遇到 `Malicious code` 的同学，请**不要在进行评测的cpp文件内写main函数**，否则评测无法正常执行。
4. **NEVER INCLUDE CPP**