**饭店安排桌位系统设计报告**

张钧尧 19307130226

环境：visual studio 2019

**一、需求分析：**

本题主要有两个对象：顾客和桌位，因此我设计了两个类Customer和Table分别表示顾客和桌位，又设计了一个结构Moment表示时间。所以本程序主要有四个模块：

Customer：定义Customer类和实现顾客信息的生成、存储、分析和输出。

Table：定义Table类和实现桌位信息的获取、储存、更新和寻找顾客要坐的桌位。

Moment：定义了Moment结构和实现了时间的加减法和比较大小。

Main：main函数。

**二、概要设计：**

**1、Moment模块：**

结构中包含两个int：hour和minute。

由于该问题中的时间跨度在24小时以内，所以比较大小时只需要以11点为界限，小于11点的一定大于等于11点的。

时间的加法只需要考虑当分钟数加起来大于等于60和小时数加起来大于等于24的特殊情况即可。

时间的减法显然本问题中使用时可以规避掉出现“负时间”的情况。需要考虑来自不同天数的时刻相减的问题，因此我将小于11点的时间的hour加上24，然后进行计算。这时只需要考虑较大时间的分钟数小于较小时间的分钟数的特殊情况，这时候计算结果的小时数要减一，分钟数要加60。

**2、Table模块**

类中包含两个量，表示桌位是否在使用的isusing和表示桌位正在被使用时使用结束时间availtime。

（一）定义了构造函数，其中默认isusing=false；定义了获取私有成员信息的接口函数gettime和getuse；定义了改变私有成员的函数changetime和changeuse。

（二）我用一个名为Tables的vector来存储桌位信息，Tables的每个元素都是一个vector，Tables[i]表示有i个座位的桌子的集合（i≥1），其中Tables[i]的索引为桌子的编号，Tables[i]的大小即为有i个座位的桌子的数目，即vector<std::vector<Table>> Tables。

（三）读取并储存桌位信息的函数为void create(vector<std::vector<Table>>& Tables)，从input\_table.txt中读取桌位信息，该文件中每一行有两个数字a和b，表示表示这种桌位有a张，每张可坐b个人。如果Tables的大小小于b+1，则扩大Tables，然后考虑Tables[b]的大小，如果小于a，则扩大Tables[b]。

（四）改变桌位信息的函数为void change(vector<vector<Table>>& Tables, Moment t)。遍历每一张桌子，如果桌子正在被使用且结束使用时间小于t，则新该桌子的isusing为false。

（五）查找顾客要坐的桌子的函数为bool gettable(vector<vector<Table>>& Tables, int n, int& x, int& y)，其中n为当前顾客的人数。该函数首先按座位数从小到大的顺序遍历座位数大于等于n的所有桌子，判断是否有能容纳下n个人的空桌子存在，如果有，结束循环，将桌子的座位数和编号分别存在x和y中，返回true。如果不存在空桌子，再次按座位数从小到大的顺序遍历座位数大于等于n的所有桌子，取结束时间最小的桌子，将桌子的座位数和编号分别存在x和y中，返回false。

**3、Customer模块：**

类中包含六个量：arrive（到达时间）、wait（等待时间）、starteat（开始用餐的时间）、eattime（吃饭时长）、leave（离开时间）和number（顾客人数）。构造函数默认eattime的hour变量为0，因为顾客的用餐时间不超过30分钟。

（一）创建顾客信息的函数为void creattxt()。规定饭店一天接待的顾客不小于100组且小于1000组，所以首先随机生成顾客的人数n（100~999）。建立三个vector：as、ahs和ams，分别储存已经生成的顾客的人数、到达时间的小时数和分钟数，然后循环n次，每次循环随机生成一组顾客的人数a、到达时间的小时数ah、到达时间的分钟数am、吃饭时长的分钟数e。规定顾客人数不超过8人，因此a的范围为1~8；为按时打烊规定饭店凌晨1点开始不接待客人，因此ah的范围为0~1、11~23，am的范围为0~59；规定每组顾客的吃饭时长应该不小于5分钟，不超过30分钟，所以e的范围为5~30。将新生成的信息和已经生成的顾客信息进行比较，如果发现已经生成的顾客中存在与新生成的顾客的人数和到达时间均相同，则循环后退一次，重新生成该组信息；如果不存在，就将新信息存进as、ahs和ams中，并输出到input\_customer.txt中。

（二）读取并存储顾客信息的函数为void create(vector<Customer>& Customers)。我是用一个vector：Customers储存顾客信息的。该函数每次读入一行，然后将顾客的人数、到达时间和吃饭时长存进Customers中。

（三）处理顾客信息的函数是void handle(vector<Customer>& Customers, vector<vector<Table>>& Tables)。该函数分为两个部分：

（1）排序：

该部分只用了一个sort函数，sort函数中定义了一个comp的函数控制顾客的排序。comp函数中规定顾客的到达时间越大顾客越大，如果到达时间相同，则人数越大顾客越大。

（2）安排桌子并计算：

该部分对排好序的Customers从头到尾进行遍历，每一次先以Tables和顾客的到达时间作为参数调用Table模块中的change函数对桌位信息进行更新。然后定义两个int：p和q，以Tables、顾客人数、p和q作为参数调用Table模块中的gettable函数，如果返回值为true，则将Tables[p][q]的isusing改为true，将该顾客的等待时间设为零，然后计算顾客的开始用餐时间和离开时间；如果返回值为false，则将Tables[p][q]的availtime加上该顾客的吃饭时长，将该顾客的等待时间设为Tables[p][q]的availtime和该顾客的到达时间的差，然后计算顾客的开始用餐时间和离开时间。

（四）输出顾客信息的函数是void out(vector<Customer>& Customers)。该函数将顾客信息输出到output\_customer.txt，然后将一天中所有组顾客在饭店的平均停留时间、平均等位时间和最后一组离开饭店的顾客的离开时间输出到output\_data.txt。该函数对处理好的Customers从头到尾进行遍历，遍历前定义变量end、waitminute和stayminute分别存储最后一组顾客离开的时间、一天中所有组顾客在饭店的总的等位时间的总和和一天中所有组顾客在饭店的总的停留时间。每次遍历先对end、waitminute和stayminute进行更新，然后按格式输出每组顾客的信息。遍历结束后计算并输出一天中所有组顾客在饭店的平均停留时间、平均等位时间和最后一组离开饭店的顾客的离开时间。

**4、main模块：**

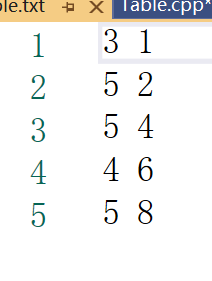
该模块调用Table模块和Customer模块中的函数进行桌子信息的读取以及顾客信息的创建、读取、处理和结果的输出。

**三、详细设计（见源文件）**

**四、调试分析：**

**1、测试数据及结果：**

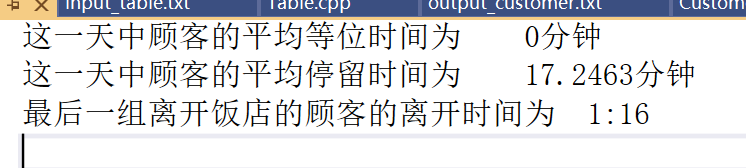
测试数据中只有桌位的信息需要自己设计，设计如下：



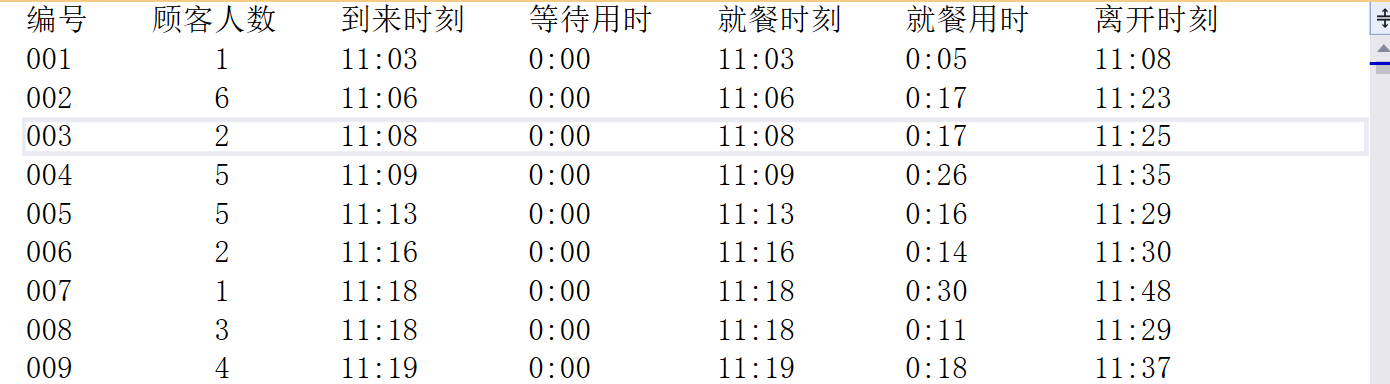
结果在output\_customer.txt和output\_data.txt中，下面是三次结果：

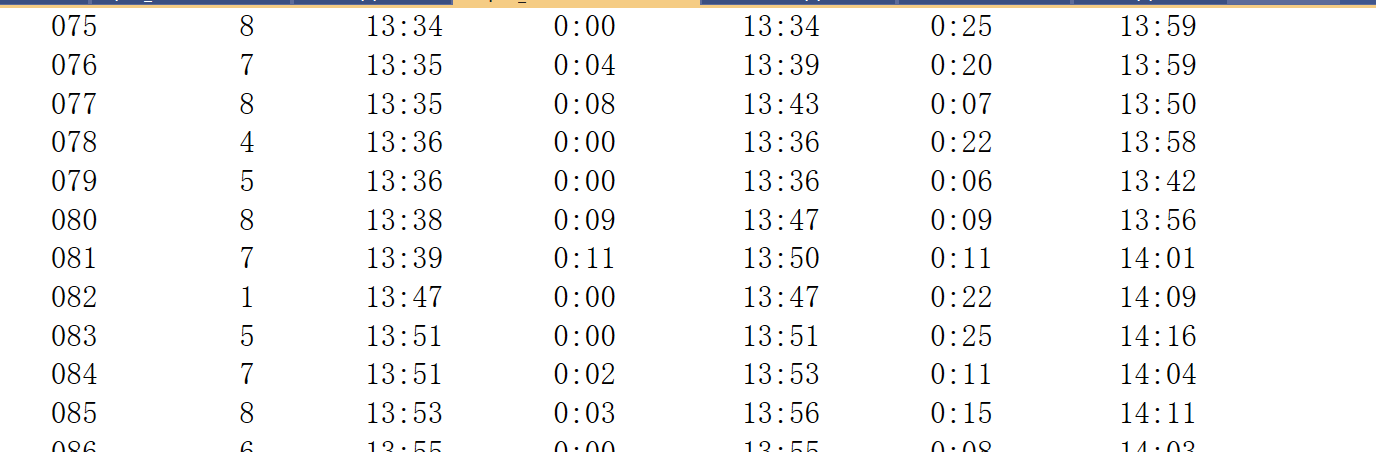
（一）

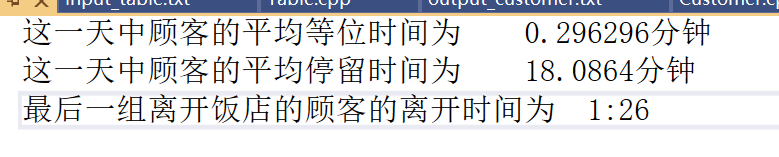




（二）

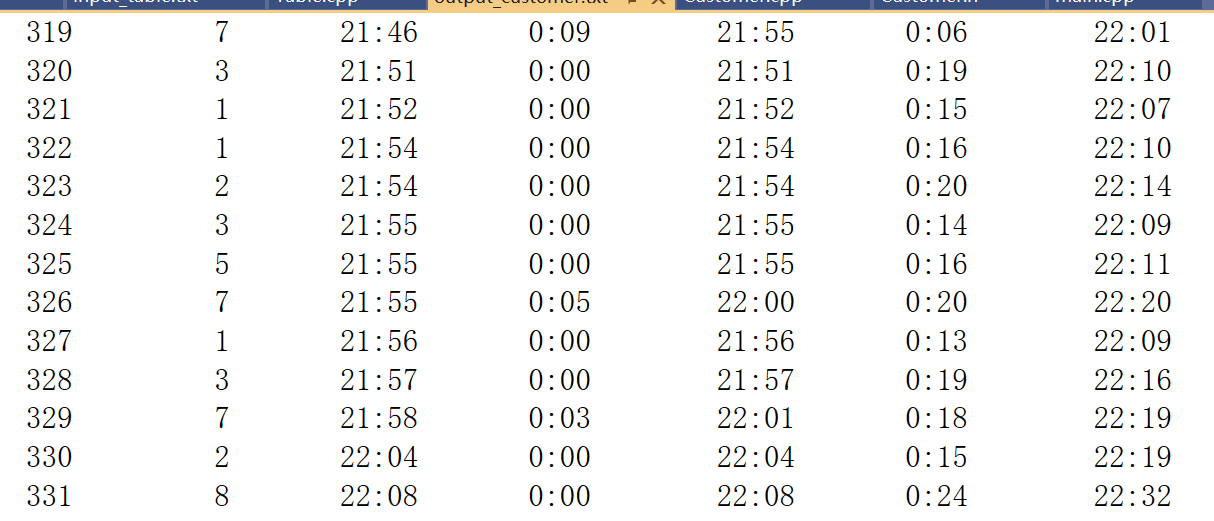


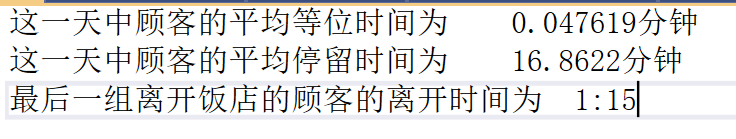




（三）







**2、时间复杂度分析：**

以下只对关键的函数进行分析：

（一）读取并储存桌位信息的函数void create(vector<std::vector<Table>>& Tables)的时间复杂度为：O(n) ，n为桌子的张数。

（二）改变桌位信息的函数void change(vector<vector<Table>>& Tables, Moment t) 的时间复杂度为：O(n) ，n为桌子的张数。

（三）对于查找顾客要坐的桌子的函数bool gettable(vector<vector<Table>>& Tables, int n, int& x, int& y)的时间复杂度为：O(m) ，m为座位数大于等于n的桌子的张数。

（四）创建顾客信息的函数void creattxt()的时间复杂度为O()，n为顾客的组数。

（五））读取并储存桌位信息的函数void create(vector<std::vector<Table>>& Tables)的时间复杂度为O(n) ，n为顾客的组数。

（六）处理顾客信息的函数void handle(vector<Customer>& Customers, vector<vector<Table>>& Tables)的时间复杂度为O(m×n)，m为桌子的张数，n为顾客的顾客的组数。

（七）输出顾客信息的函数void out(vector<Customer>& Customers)的时间复杂度为O(n)，n为顾客的组数。

**3、存在的问题及改进方案：**

（一）只能通过控制桌位信息和提早不接待新顾客的方式控制饭店能在凌晨2点之前正常关门，一旦将桌子的张数减少，就有可能出现顾客等待时间过长或最后离开的顾客的离开时间超过凌晨2点的现象；但如果桌位过多，顾客等待时间不长，就会出现凌晨1点半左右就没有生意的现象，影响饭店收入。

改进方案：延后不再接待新的顾客的时间，如果预计顾客的离开时间可能超过凌晨2点，通过催促顾客或给顾客打包的方式让顾客在2点离开。程序表现为如果顾客的离开时间大于2:00就设为2:00，并重新计算其他信息。

（二）本程序中各个人数的顾客组出现的概率是相同的，但现实情况不是这样。

改进方案：对一般的饭店中顾客的人数进行调查，按照各个人数的顾客出现的概率生成顾客的信息。

（三）本程序中各个时间点顾客出现的概率是相同，但显然饭点顾客会更多，如果是吃火锅或烧烤一类的饭店晚上顾客会更多。

改进方案：针对一种类型的饭店进行调查，按各个时间顾客出现的概论生成顾客的信息。

**五、课程设计总结：**

在本次设计作业中，我充分体会到了使用正确的数据结构对程序的重要性。使用正确的数据结构不仅能增强程序的可读性和条理性，而且能大幅度优化程序的时间复杂度和空间复杂度。

同时我认识到程序设计必须结合世纪问题，根据实际情况和实际需求来设计程序。这样在设计程序时会更有针对性，不仅会增强程序的实用性，而且会避免一些不必要的工作的出现。

最后，之前的学习中我一直没有重视程序的运行的效率，但是这次的程序设计中我就遇到了时间复杂度过大程序运行时间过长的问题。因此设计程序时一定要有意识地在保证程序能正确稳定运行的前提下采用效率更高的算法，在内存足够的情况下可以牺牲空间优化时间。

**六、参考资料：**

对于读取文件时如何判断文件结束的问题我上网查了一下，参考了下面这个网站的方法：<https://blog.csdn.net/botao_900422/article/details/8733709>。