# 双电梯程序说明文档

***1.*** ***成员组成以及工作分配：***

**成员组成：**

**工作分配：**

***2. 程序实现功能：***

输入来的用户的进入楼层、出楼层、最大容忍时间、选的双电梯的序号以及下一位用户来的时间，确定并输出两个电梯的运行流程包括上升、下降、进入、离开、去起始层、停留、开门、关门等等。

***3. 程序实现原理以及基本框架：***

**核心思想**：通过离散的时钟事件表去模拟电梯的各个活动

**数据结构使用**：主要为链表（包含类似图存储结构邻接表的组合链表）

* 事件链表activity：通过动态链表的方式按时间顺序把各活动（包含该活动的时间和操作信息）连接起来
* 排队序列WaitQueue：建一个头结点数组WaitQueue[MAX\_FLOOR\_NUM]来存储各楼层排队等待序列的第一个人的指针，然后采取动态链表的方式一个一个插入到队伍后边，来确定当前层有无人要进电梯
* 电梯运行时当前楼层的电梯出的人的序列elevator：和WaitQueue类似，采取和图的邻接表的结构相似的组合链表方式去存储数据，由于双电梯的存在，分了两个elevator以便确定当前电梯内该楼层有无人要出序列
* 还有很多全局变量（关于电梯的状态信息等），用于各函数间更新信息，如Floor1,Floor2,Count,Time,State1,State2为电梯1、2所处楼层，用户序数，当前电梯所处离散时间，电梯1、2的状态等等
* 表示当前电梯的用户请求信息：

CallUp[1..MAX\_FLOOR\_NUM] , CallDown[1..MAX\_FLOOR\_NUM] ,CallCar[1..MAX\_FLOOR\_NUM],若值为1分别表示当前层有上升请求和下降请求以及有到该目标层的请求

* 同时还有各种代表电梯状态和电梯各种操作所需时间数的宏定义：

UpTime,InOutTime,GoingUp等等

**分函数即基本操作使用**： 因数据结构以链表为主，因此函数主要以动态链表的删除和插入操作为主

* timer()：根据串联起来的事件activity动态链表并按时间的顺序执行此刻两个电梯需要执行的小操作，主要是对activity链表的结点的删除操作以及决定此刻该调用哪个离散时间对应的小函数
* acter()：将某个时刻要执行的操作作为一个事件插入到时间表里，该事件包含数据有该事件的事件、要执行的电梯序号以及事件及操作所对应的函数指针，因此主要操作为对acitivity动态链表的插入操作
* controller()：根据不同楼层的请求来确定双电梯运动方向以及是否开门的电梯调度控制函数，若当前层有请求则开门，若当前层无则判断其他层有无请求，调度思想为：若电梯正在向上运行，则上层有请求时继续向上把这些请求完成，若上层已无请求而下层有请求则向下，若上下均无请求则停留，然后若长时间停靠则去电梯的起始楼层；若电梯正在向下运行，则下层有请求时继续向下运行，若下层已无请求而上层有请求则向上，若上下均无请求则停留，然后若长时间停靠则去该电梯的起始楼层，其中根据上下是否有请求判断电梯方向的子函数为OtherFloor()。该函数中只涉及不同条件的函数调用，不存在链表的插入删除操作等。
* OtherFloor()：作为controller的子函数去判断其他层是否有请求以及给出方向的函数。
* manager()：人员去留的管理函数，每个用户都有一个最大容忍时间，若全部楼层某一层的等待队列中有任何一个用户等待时间超过这一时间则要作为队头出队列，则该函数所执行的主要操作为WaitQueue这一邻接表中的不同楼层的队列的结点的删除操作。
* ComePeople()：为用户到来的函数用来输入用户的相关信息，以及更新两个电梯的请求变量的数据，并为到来的下一个用户添加acter。
* Up()：为电梯的向上操作，基本执行流程为：若上升一层后该层有请求则为停靠函数Idle()添加acter，停靠的同时用controller来同步开门，若无请求则一直接着为Up添加acter直至顶层。
* Down()：为电梯的向下操作，基本执行流程为：若下降一层后该层有请求则为停靠函数添加acter，停靠的同时用的是controller来同步开门，若无请求则一直接着为Down()添加acter直至底层。
* In()：为电梯开门后用户进入电梯的操作，除了显示用户进入电梯的信息之外，还涉及WaitQueue该层用户的结点的删除，以及elevator该层用户的结点的插入，并更新该层该用户的目标层CallCar为1，还有若当前层WaitQueue仍有用户则继续为In()添加acter。
* Out()：为电梯开门后用户出电梯的操作，除了显示用户出电梯的信息之外，还涉及elevator该层用户的节点的删除，还有若当前层WaitQueue还有用户要进或者elevator还有用户要出则继续为In()和Out()添加acter。
* IfInOut()：为OpenDoor()开门后内部执行的每隔一段时间判断是否还有人进出的函数，若无人进出，则为CloseDoor添加acter，否则继续为IfInOut()添加acter。
* OpenDoor()：为电梯的开门操作，由controller根据当前层是否有人进出选择去调用，开门后除显示信息之外，还有每隔一段时间判断是否还有人进出即为IfInOut添加acter，还有开门后若WaitQueue该层不为空则入为In()添加acter，若elevator该层不为空则出为Out()添加acter。
* CloseDoor()：为电梯的关门操作，显示电梯关门信息以及电梯的状态更新。
* ToIdle()：电梯停靠操作，显示电梯在某层停靠。
* ToFirstIdle()：因长时间停留在某一层而无请求则要回到初始楼层候命。

**程序大致运行流程**：开始先初始化一些信息，入电梯1起始楼层为1，电梯2起始楼层为5，同时为一些变量赋初值为0，然后先输入第一位用户信息，紧接着开始调用timer()去执行真个用户到来和电梯运行流程，电梯1和电梯2共用一个CallCar和Time以及activity即使用一个控制流程，两个电梯的内部实现完全一样，只是该程序中根据用户的选择在分别调用两个电梯，但是当用户选择了电梯1时电梯2仍会在该层停靠只是不会开门。例子：来了一位选了电梯1的用户大概各个函数调用情况如下：先为下一位用户的到来添加acter，然后电梯1去该用户的入楼层去开始Up，直到该层有请求时即到了出楼层时开始为Idle()添加acter，然后同时为OpenDoor()添加acter，在到OpenDoor()调用时还会为IfInOut添加acter，然后为In()添加acter，然后为CloseDoor添加acter，这位一个用户的整体请求以及满足流程，其中若在未到达出楼层时出现一个用户的请求则停靠把该用户接入并继续上升或下降。当整个程序运行至Time超过极限值时程序运行结束。

***4. 程序使用和运行说明：***

该程序为单文件程序，不涉及项目组成，因此使用任何C编辑器编译运行即可。

**程序输入：**即为按照输出提示去输入即可，注意楼层仅包含1至5层，且下一位用户到来时间尽量控制在100内且要随机，才能更好的模仿电梯的运行情况。

**程序输出：**在一位用户到来和该用户信息输入完成后会有提示符去划分，因此可以很清楚的看到一位用户的到来的信息输入开始的地方，其余输出即为电梯1和2的调度过程和用户的进出过程。

***5. 程序运行结果：***



