Level0-0

首先安装visual studio。曾经学习c++使用过devc++，但腻了（bushi）；想尝试使用vscode，但是恰逢有舍友使用vs，就以老带新，装了vs，还省去了自己配置环境、编译器的步骤。熟悉了一下创建项目和新建代码，遇到的第一个问题是vs不能用scanf？了解情况后选择用define+scanf\_s给他盖掉。感觉c和c++的语法基本相通，没什么不适感，浅尝输出hello world结束了安装阶段。

（更新：用#define \_CRT\_SECURE\_NO\_WARNINGS解决scanf，不然别的平台跑不了力）

level0-1

学习使用git，选择github。因为steam的缘故，有Watt Toolkit，挂起加速来也十分稳定。通过b站教学视频，掌握新建远程仓库的方法，准备在完成level1-1后，再尝试上传文件。

（后续：github网页版，只要创建readme就可以拖拽上传）

lever0-2

已具备一定的代码实现能力

学习手动编译运行文件（学习命令行）笔记如下：

不能识别“gcc”

配置环境变量：我下载过dev，找到文件所在位置，bin的路径拿去配置

gcc hello.c -o hello.exe 表示编译运行，生成可执行文件

可分为编译和链接部分

gcc -c hello.c 表示只编译不链接，生成.o文件，即二进制文件

gcc hello.o -o hello.exe 表示只链接

编译又可以分为预处理文件、汇编文件、目标文件（.o）三个阶段

gcc -E hello.c -o hello.i（一坨，路径）

gcc -S hello.c -o hello.s

其他补充：cd 文件路径 dir 查看当前文件夹

如果文件名带空格，那么在命令行中还要用双引号括起来

如果不设置“-o” 则文件名为a

（然后就是审题时，被这个只有呼叫键，没有上下键整的很迷茫，假如我按照有上下键来做，就当做我运气比较好：在上行时接到上楼的、下行时接到下楼的吧（笑）。如果只有呼叫键，其实实现方法相同，只不过有部分人经历掉头环节，然后只要经过有人楼层考虑接上就好了）

lever1-1

一人很简单，电梯去接他，然后去目的地。特判一下起点是否一致即可。

（思考不止超过四人情况）考虑到电梯其实就是起始点往返（除了开始时要前往起点），直接每次载小于等于4人，不断累加用时即可

level1-3

1、注意：每个⼈只有进⼊电梯后才会按下他们的⽬的楼层

2、不同楼层的⼈同时按下电梯⻔的按键

说明不能上帝视角安排最优解（降低了难度:D），同一起点但不同终点的人上电梯是随机的。我的策略是电梯来回动（像刷子一样），刷的幅度是从大到小，即先接低楼层上行的和高楼层下行的，途中有空位可以顺带上客。

另一种想法是先接低楼层下行的和高楼层上行的（这样可能总体排队时长较短？因为排队时让更快的人先办事其实高效，没有细想如何实现）

想法源于实际，电梯一般尽可能保证同向运动吧。

实现阶段：考虑到只有十层楼，直接采用模拟，暗线是电梯在1-10往返，明线是只输出停靠楼层，这样比如我只需要上到8楼，8会输出一次，而电梯悄咪咪跑到十楼掉头你也不知道（实在是懒狗，能简单实现就简单实现）

细节是先下后上，不仅文明，而且保证尽可能上客，不让电梯空间浪费。

level2-1

（按照有上下键并且每层不止一个人来写，感觉这样实际一丢丢）

1、绝对的公平：先到先得，坏处是低效率；

2、维持上下往返动模式，顺路能接就接，坏处是对高楼层往上、低楼层往下可能等到饿死（如一楼很多人想去高楼层，那二楼上行人就完蛋）；

所以我想混合一下，公平方面，模糊时间，如时刻1-5都视作时段1，然后完成一个时段再处理下一个时段，时段内依然维持往返动模式，试图保证效率和稳定性。

（可能有一点点分块的思想？口胡）

但我随即想到一个问题，比如8楼有时段1、2的人在等，8楼停靠了，不可能只让时段1的上，一定都会蜂拥而上，也就是说时段2人吃了时段1人的红利（笑），当然，同一层楼肯定是先到先得。

习惯自己写swap、cmp，一开始按照c++的“int &a”来引用，发现报错，仔细学习后发现c语言的指针、引用更加繁琐

发现电梯“闲时静止不动”还挺麻烦的，需要去判断实时是否继续动，而不能去“预判”下一个呼叫人，果然联系实际就是不得不浇灭过于理想的东西qwq。

level2-2

首先代码实现上有区别，如果是一台电梯，只需要一次for循环，就可以解决一次单向运动，“但是两台电梯，为了保证时间一样，我决定一台动一步。"（初始想法）

思路上，如何调度两台，我思考了一段时间，最简单的想法应该是两台电梯都单独运作，但是比如一台1楼，一台2楼，五楼有请求，就会变成都上行，双倍功耗。然后是想尽量一台向上，一台向下，但感觉很难写，除非我强制他们来回动不停，这对高峰时段应该还行，但是冷清时段就会看起来十分愚蠢。然后试图空想出分配给两个电梯的方案，决定上网学习，发现过于高深（都涉及大数据、神经元网络啥啥），且针对不同需求有不同写法，如优先的楼层？上or下高峰？已启动电梯优先？

最后，敲定中庸的算法：一台服务奇数层，一台服务偶数层。

服务奇数层指奇数层离开电梯，奇偶层都可以上电梯；（这在现实中是可以实现的，任何楼层都有两个电梯的门，注明一个奇数落客，一个偶数，在电梯箱内只有这种楼层可以按亮）

不足：

1、在低强度下，如只有1楼两人，分别要上到3、4，就要动用两台电梯。但考虑到对于教学楼，使用电梯集中在高峰，可以满载，故不足忽略。

2、对于教学楼，一楼的人流一定较大，所以应该两台电梯都允许前往一楼，（但是这样代码就不好写了）

一点狡辩：上楼比较累，反正在一楼两台电梯都可以呼叫，所以下楼到不了一楼也没事吧（捂嘴笑），下到2楼再走楼梯怎么了doge

实现过程：

1、也可以不用一台动一步，我先把0号梯跑完，再把1号梯跑完，输出先存起来，最好按时间排序输出，天衣无缝