学习文档

声明：本人在完成本次任务时，所有代码均为个人独立编写，未以复制，换皮或任何其他形式从互联网，他人处获取代码。所有内容均无 AI 生成。

在完成本次招新测试任务的过程中，以下几点是我认为值得记录的。

1. git 的使用，使用 git 时，应注意先 init 仓库，再添加文件，并且 add 命令只会将文件添加至暂存区，需要 commit 命令（-m 后的参数如果省略，commit 命令将会失败）。添加完成后，可以使用 git ls-files 及 git status 查看仓库内文件和未包含的文件。
2. 编程过程中，相比代码的执行效率，代码的可读性和易维护性更为重要（在小规模情形下）。容易读懂的代码未必速度慢，但不容易读懂的代码几乎必定出现大量奇形怪状的 bug，增加调试成本。
3. 将常量数字使用#define 定义符号常量更方便。
4. 应当尽量将执行类似功能的代码放在一起，更容易维护。
5. 在查找电梯算法时，最初认为 SCAN，LOOK 算法都很粗糙而不满意，试图通过加权，赋分等形式让机器在遍历可能解法后选择分数期望最小的方法，结果发现在人数较多的情况下的推演根本不现实……后来在网上找到应用了神经网络，模糊控制啥的算法，又看不懂……

以下是 level1.3 程序的一点说明:

名称	默认值	含义
MAX_CLIENT	100	最大识别的客人数量
MAX_CAPABILITY	4	电梯最大载客量
NUM_FLOOR	20	学院楼层数
MAX_CHARINPUT	100	test2.txt 文本中一行最大字符数量
MAX_TRAIL	1000	电梯轨迹保存最大条数
MAX_SHOWINLINE	15	电梯轨迹一行展示数量

通过这些符号常数，能够更好地调试程序

程序主体算法:

读取文件->为文件数据排序->电梯运行

电梯运行算法（循环）:

计算客人进入->输出电梯状况->选定电梯运行目标-> 判断循环是否终止->移动电梯->计算客人离开

```
int flag=target[0];
int flag2 =infloor[0];
for(int i=0;i<number;i++)
    if(infloor[i]!=0)
    {
        flag2=infloor[i];
        break;
    }
if(flag2==0)
    return;
if(!currentWay)
{
    for(int i=1;i<currentCilentNum;i++)
        if((flag<currentFloor)||((target[i]>currentFloor)&&(target[i]<flag)))
            flag=target[i];
    if(flag<currentFloor)
        for(int i=0;i<currentCilentNum;i++)
            if(target[i]>flag)
                flag=target[i];

    if(currentCilentNum>=MAX_CAPABILITY)
        currentTarget=flag;
    else{
        //int flag2=infloor[0];
        for(int i=1;i<number;i++)
            if(infloor[i]!=0)
                if(((flag2<currentFloor)||((infloor[i]>currentFloor)&&(infloor[i]<flag2)))&&(infloor[i]!=currentFloor))
                    flag2=infloor[i];
        if(currentCilentNum>0)
            if(flag2<currentFloor)
            {
                if(flag<=currentFloor)
                {
                    for(int i=0;i<number;i++)//////////
                        if(infloor[i]!=0)
                            if(infloor[i]>flag2)
                                flag2=infloor[i];
                    currentTarget=max(flag,flag2);
                    currentWay=!currentWay;
                }
                else
                    currentTarget=flag;
            }
            else if(flag2==currentFloor)
            {
                currentTarget=flag;
            }
            else if(flag<=currentFloor)
                currentTarget=flag2;
            else
                currentTarget=min(flag,flag2);
        else
        {
            if(flag2<currentFloor)
            {
                for(int i=0;i<number;i++)
                    if(infloor[i]!=0)
                        if(infloor[i]>flag2)
                            flag2=infloor[i];
                currentTarget=flag2;
                currentWay=!currentWay;
            }
            else if(flag2==currentFloor)
            {
                //currentTarget=flag;
                printf("Over\n");
                return;
            }
            else{
                currentTarget=flag2;
            }
        }
    }
}
else{
    for(int i=1;i<currentCilentNum;i++)
        if((flag>currentFloor)||((target[i]<currentFloor)&&(target[i]>flag)))
            flag=target[i];
    if(flag>currentFloor)
        for(int i=0;i<currentCilentNum;i++)
            if(target[i]<flag)
                flag=target[i];

    if(currentCilentNum>=MAX_CAPABILITY)
        currentTarget=flag;
    else{
        //int flag2=infloor[0];
        for(int i=1;i<number;i++)
            if(infloor[i]!=0)
                if(((flag2>currentFloor)||((infloor[i]<currentFloor)&&(infloor[i]>flag2)))&&(infloor[i]!=currentFloor))
                    flag2=infloor[i];
        if(currentCilentNum>0)
            if(flag2>currentFloor)
            {
                if(flag>currentFloor)
                {
                    for(int i=0;i<number;i++)
                        if(infloor[i]<flag2)
                            flag2=infloor[i];
                    currentTarget=min(flag,flag2);
                    currentWay=!currentWay;
                }
                else
                    currentTarget=flag;
            }
            else if(flag2==currentFloor)
            {
                currentTarget=flag;
            }
            else if(flag>=currentFloor)
                currentTarget=flag2;
            else
                currentTarget=max(flag,flag2);
        else
        {
            if(flag2>currentFloor)
            {
                for(int i=0;i<number;i++)
                    if(infloor[i]<flag2)
                        flag2=infloor[i];
                currentTarget=flag2;
                currentWay=!currentWay;
            }
            else if(flag2==currentFloor)
            {
                //currentTarget=flag;
                printf("Over\n");
                return;
            }
            else{
                currentTarget=flag2;
            }
        }
    }
}
```

经过不断优化，主题代码由百余行缩短为 70+行，（上图为修改前主体代码）代码反而更加容易维护，调试，bug 也容易被发现（下图为修改后代码）。

```
int in_goal_up=0;
int in_goal_down=NUM_FLOOR+1;
int out_goal_up=0;
int out_goal_down=NUM_FLOOR+1;

for(int i=0;i<currentClientNum;i++)
    if((in_goal_up<currentFloor)||((target[i]>currentFloor)&&(target[i]<in_goal_up)))
        in_goal_up=target[i];
if(in_goal_up<currentFloor)
    in_goal_up=0;
for(int i=0;i<currentClientNum;i++)
    if((in_goal_down>currentFloor)||((target[i]<currentFloor)&&(target[i]>in_goal_down)))
        in_goal_down=target[i];
if(in_goal_down>currentFloor)
    in_goal_down=0;
for(int i=0;i<number;i++)
    if((infloor[i]>0)&&(infloor[i]!=currentFloor))
        if((out_goal_up<currentFloor)||((target[i]>currentFloor)&&(infloor[i]<out_goal_up)))
            out_goal_up=infloor[i];
if(out_goal_up<currentFloor)
    out_goal_up=0;
for(int i=0;i<number;i++)
    if((infloor[i]>0)&&(infloor[i]!=currentFloor))
        if((out_goal_down>currentFloor)||((target[i]<currentFloor)&&(infloor[i]>out_goal_down)))
            out_goal_down=infloor[i];
if(out_goal_down>currentFloor)
    out_goal_down=0;


if(!(in_goal_up||in_goal_down||out_goal_up||out_goal_down))//determine to return or not
    return totaltime;

if(currentClientNum>=MAX_CAPABILITY)
{
    if(!currentWay)
        currentTarget=in_goal_up?in_goal_up:in_goal_down;
    else
        currentTarget=in_goal_down?in_goal_down:in_goal_up;
}
else{
    if(!currentWay)
    {
        if(out_goal_up||in_goal_up)
        {
            if(out_goal_up&&in_goal_up)
                currentTarget=min(out_goal_up,in_goal_up);
            else
                currentTarget=out_goal_up+in_goal_up;
        }
        else{
            if(out_goal_down&&in_goal_down)
                currentTarget=max(out_goal_down,in_goal_down);
            else
                currentTarget=out_goal_down+in_goal_down;
        }
    }
    else{
        if(out_goal_down||in_goal_down)
        {
            if(out_goal_down&&in_goal_down)
                currentTarget=max(out_goal_down,in_goal_down);
            else
                currentTarget=out_goal_down+in_goal_down;
        }
        else{
            if(out_goal_up&&in_goal_up)
                currentTarget=min(out_goal_up,in_goal_up);
            else
                currentTarget=out_goal_up+in_goal_up;
        }
    }
}
```

最后，感谢 Dian 团队的本次任务，让我从中收获不少快乐，也重新体会到被 bug 支配的恐惧……

Reference

https://www.cnblogs.com/jianyungsun/archive/2011/03/16/1986439.html
https://blog.csdn.net/qq_43265673/article/details/107392047
https://blog.csdn.net/u_7890/article/details/81565679
https://www.runoob.com/cprogramming/c-function-sscanf.html