# 学习文档

额外的学习经历：学会了gpt注册并且成功使用了gpt。

1.C语言学习过程函数return返回主调函数

git 学习过程中找不到本地库在哪里。

2..gcc 编译，第一个是配环境变量，不知道要tdm64还是mingw w64，发现mingw w64安装失败，只能装tdm64。打开cmd后无法跳到文件夹，解决方式为直接在文件夹用终端打开，后面研究发现是斜杠反了。Dir可以查看所有文件，挺好用的。（该部分详细内容见gcc编译学习）

3.文件路径问题；

1、文件在当前目录（以图像文件为例，当前项目文件为中心）

"./1.jpg" 或 "1.jpg"

2、文件在上层目录

（1）在上层目录下

"../1.jpg"

（2）在上层目录下的一个Image文件夹下

"../Image/1.jpg"

（3）在上上层目录下

"../../1.jpg"

3、文件在下一层目录(Image1文件夹)

"./Image1/1.jpg"

4、根目录表示法,任何页面访问Image下的Image.jpg图片

"C:/Image/1.jpg"

A,B,C楼层判断大小问题

解决方案一：直接用if判断大小，但是很麻烦；

方案弃用，太蠢了这个方法。

作业调度算法，最简单的应该是遍历楼层。

一开始1-1和1-2用了很蠢的条件判断，发现根本没必要，直接用abs函数就可以了.。。

1-1：

方法一：采取if语句。

方法二：直接abs函数就好了（我是傻逼）。

1-2：

文件读取

Fopen

|  |  |
| --- | --- |
| "r" | 以“只读”方式打开文件。只允许读取，不允许写入。文件必须存在，否则打开失败。 |
| "w" | 以“写入”方式打开文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么清空文件内容（相当于删除原文件，再创建一个新文件）。 |
| "a" | 以“追加”方式打开文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么将写入的数据追加到文件的末尾（文件原有的内容保留）。 |
| "r+" | 以“读写”方式打开文件。既可以读取也可以写入，也就是随意更新文件。文件必须存在，否则打开失败。 |
| "w+" | 以“写入/更新”方式打开文件，相当于w和r+叠加的效果。既可以读取也可以写入，也就是随意更新文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么清空文件内容（相当于删除原文件，再创建一个新文件）。 |
| "a+" | 以“追加/更新”方式打开文件，相当于a和r+叠加的效果。既可以读取也可以写入，也就是随意更新文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么将写入的数据追加到文件的末尾（文件原有的内容保留）。 |
| "t" | 文本文件。如果不写，默认为"t"。可以不写 |
| "b" | 二进制文件。可以不写 |

（摘自网上）

文件一旦使用完毕，应该用 fclose() 函数把文件关闭，以释放相关资源，避免数据丢失。fclose() 的用法为：

int fclose(FILE \*fp);

fp 为文件指针。例如：

fclose(fp);

文件正常关闭时，fclose() 的返回值为0，如果返回非零值则表示有错误发生。

读取文件，首先定义一个FILE指针，初始化这个指针 =null。

然后fopen打开文件，file要加取地址符，后面写文件名，然后再加上面所述的r或者w之类的。

接下来就是f函数，包括fscanf，fclose，fgets，fread之类的。。。、

本题运用了fscanf函数读取。。。。。？？？？？怎么报错了？原来是vs的问题

数据流

就C程序而言，从程序移进，移出字节，这种字节流就叫做流。程序与数据的交互是以流的形式进行的。进行C语言文件的读写时，都会先进行“打开文件”操作，这个操作就是在打开数据流，而“关闭文件”操作就是关闭数据流。

缓冲区

在程序执行时，所提供的额外内存，可用来暂时存放准备执行的数据。它的设置是为了提高存取效率，因为内存的存取速度比磁盘驱动器快得多。

当使用标准I/O函数(包含在头文件stdio.h中)时，系统会自动设置缓冲区，并通过数据流来读写文件。当进行文件读取时，是先打开数据流，将磁盘上的文件信息拷贝到缓冲区内，然后程序再从缓冲区中读取所需数据。事实上，当写入文件时，并不会马上写入磁盘中，而是先写入缓冲区，只有在缓冲区已满或“关闭文件”时，才会将数据写入磁盘。

文件类型

文本文件和二进制文件：

文本文件是以字符编码的方式进行保存的。

二进制文件将内存中的数据原封不动的进行保存，适用于非字符为主的数据。其实，所有的数据都可以算是二进制文件。二进制文件的优点在于存取速度快，占用空间小。

文件存取方式

顺序存取方式和随机存取方式：

顺序存取就是从上往下，一笔一笔读取文件的内容。写入数据时，将数据附加在文件的末尾。这种存取方式常用于文本文件。

随机存取方式多半以二进制文件为主。它会以一个完整的单位来进行数据的读取和写入，通常以结构为单位。

1-3采取了先向下到底再向上。对于超载采取了引入了一个cache变量。如何表示人的状态？发现可以在结构体内加一个元素，该元素等于0表示还没上电梯，等于1表示上了电梯还没出来，2表示已经出来了，每次先楼层扫描循环，然后在结构体数组里面再进行一次遍历，找到可以上电梯的和可以下电梯的。解决了问题.

2-1 SCAN-EDF（这个方法太难了）创造一个结构体数组，先判断电梯方向，这边会采取一个direction函数判断方向。然后再引入一个peoplescan函数，有人进去，则第一结构体数组的该结构体转入第二结构体数组。由于出现了超载问题，还加了一个moveon函数。

问题：参数怎么在函数之间传递？公共变量可以解决。（公共时间为变量）

写了一下，现在出现几个问题：1.malloc可变结构体数组大小.2.不知道怎么在不同函数里调用同样的参数，并且每次调用后这个参数都会被修改，而不是被清除。后面发现有公共变量这个东西。3.一些语法错误。4.怎么把一个数组内的某个数据清除，memset

啊啊这种方法太蠢了受不了了

采取另一种方法：依据时间进行排序：排序之后判断初始方向（这是容易的），然后要顺带上时间允许的在路上的人，这个用一个if语句判断，每次的start\_pos和都会被更新，然后一直循环下去。

最难的地方在于方向的改变怎么办？路径上带人怎么办？