

数据结构课程设计

——校园导览系统

**各模块设计说明**

成员：董晓雨 2019211589

张明昱2019211590

邵逸辰 2019211591

班 级：2019211314

完成时间：2021/6/9

# 一、模块划分

**1、初始化模块**

初始化地图，初始化用户信息,初始化日志，初始化时间线程。

**2、主程序模块**

接收键盘键入命令，分析该命令并调用相应的模块，并以时间为轴向前推进。

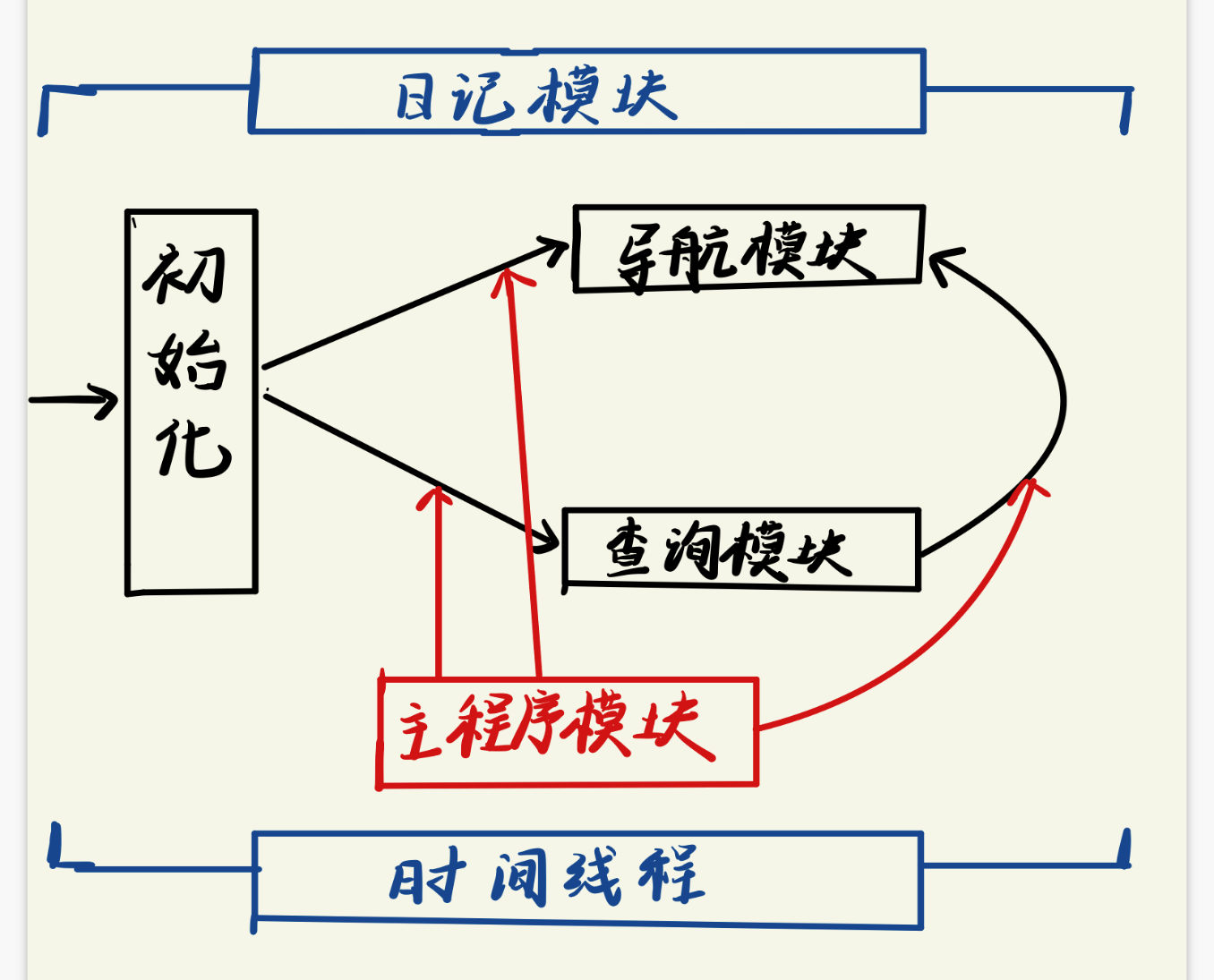
**3、查询模块**查询当前位置或指定位置周围的建筑物和服务设施。

**4、导航模块**生成相应的导航线路，并以文字形式输出到界面上。

**5、日志文件处理模块**完成相应的日志文件写入和查询结果输出等功能。

**6、时间模块**

# 二、模块调用关系



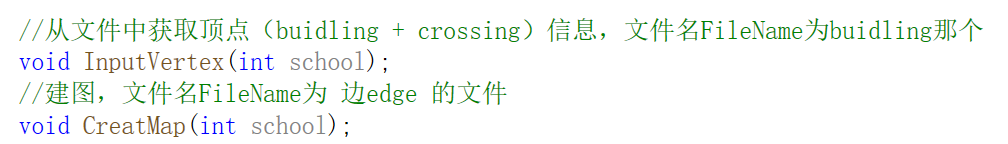
# 三、各模块详解

**1、初始化模块**

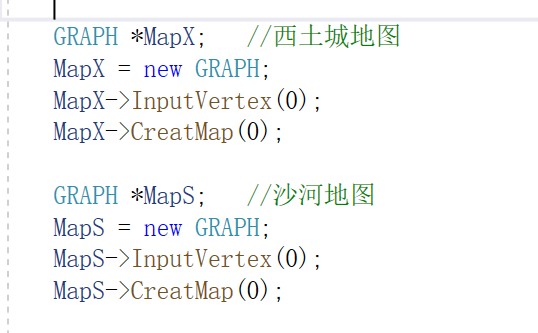
初始化模块完成地图、用户、日志和时间线程四方面的初始化。

①**地图的初始化**通过GRAPH类及其内部功能函数实现。

GRAPH类利用C++面向对象的属性和其封装性的特点，把与地图操作有关的成员变量和成员函数包裹组成地图类，其中的成员变量包括地图结点链表以及地图边链表，成员变量包括用于创建地图的void InputVertex（）和void CreatMap（）：函数通过读入固定格式的文件，以内存中储存的相应方式逐项读入内存中，首先获取信息，然后依照信息创建graph。



基于GRAPH类的良好基础，在主函数中实现地图的初始化时，只需利用动态分配简单建立两个graph类变量，并调用类中的创建函数即可轻松创建西土城地图以及沙河地图。

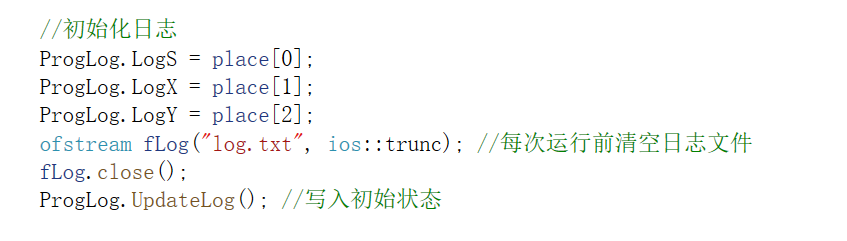


②**用户的初始化**信息依靠用户的输入交互获得信息。

对用户进行初始化时，一来需要初始化用户的地理位置为导航和查询的进行提供基础，二来还需要初始化用户的个人信息，即班级信息，以正确导入课表，给用户提供课表和考表的逻辑导航。

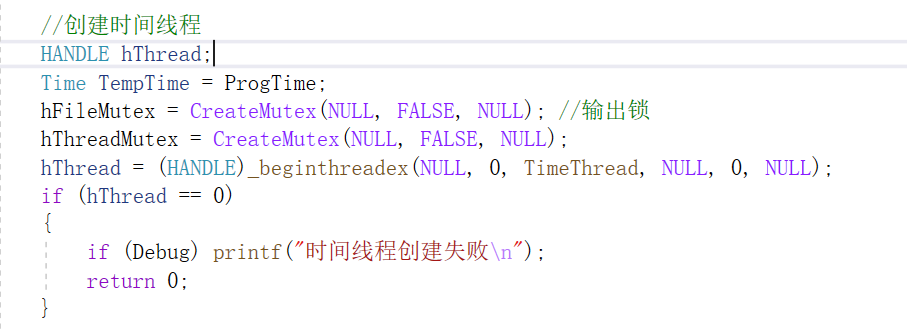
此处的初始化通过void Initial(int\* school, string\* posStart, string\* Class, int place[3])函数完成，校区信息、班级信息以用户输入获得，初始地理位置由系统默认，但会告知用户初始化位置，并允许用户输入任意一个地点的物理位置进行修改。在用户输入的地方都设置有差错控制，保证用户输入格式正确，增强程序的健壮性。

**③日志的初始化**直接通过程序内置信息写入

****

**④时间线程的初始化**

在程序开始时创建时间线程，并创建出时间锁，保证在程序接下来的运行导航和模拟行走部分时间的正常运行。

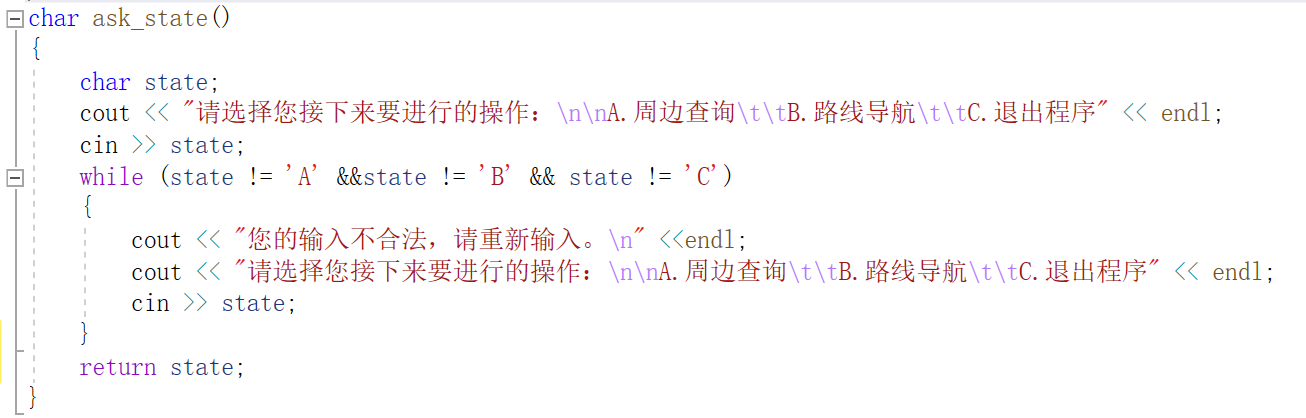


**2、主程序模块**

主程序模块接收键盘键入命令，分析该命令并调用相应的模块，并以时间为轴向前推进，配合上时间线程，是程序运行推进的骨骼。

每一次针对用户操作的询问都使用选项方式提供，并配合以差错控制，使得输入可控。同时将这些繁琐而没有太大技术价值的询问封装为函数隐藏起来，增加了程序的美观性。

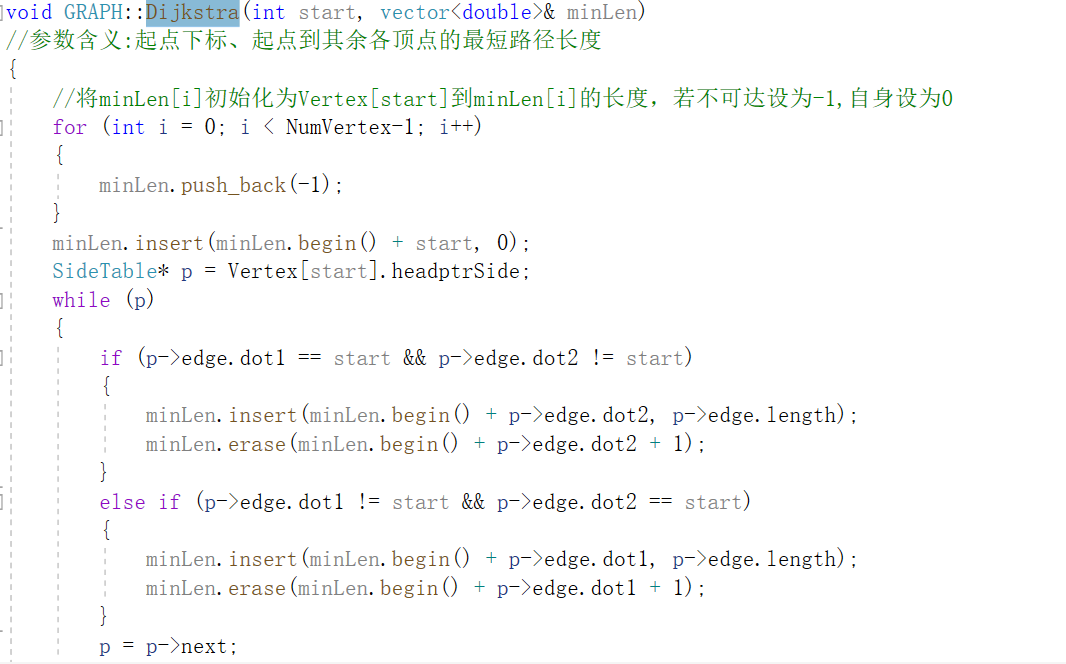
下图是一个封装起来的询问函数的实例：



**3、查询模块**

查询模块可以查询当前位置或指定位置周围的建筑物和服务设施，在确定需要查询的位置和所在校区后，传参并用void OutSearch(int place[3], GRAPH\* Map)函数输出中心地点，并输出周围distance（常量定义）范围内的建筑物信息.

内部算法为：输出中心地点时，遍历顶点表，找到 x==place[1] && y==place[2] 的dot，输出物理地址PlacePhy；输出周边时，遍历顶点表，找出距离 <= distance 的点输出，其中确认中心地点到周围各点的最短距离时，使用graph类中定义的**迪杰斯特拉算法**计算确定，算法具体实现如下图：





1. **导航模块**导航模块生成相应的导航线路，并以文字形式输出到界面上。

用户选择进入导航系统后，系统先通过询问用户确认路线是否跨校区，是否要使用自行车，是否需要途径多点。根据这些选择为用户匹配不同的操作。在进行完导航方式的选择后，提示用户输入起始点和终点，此时要对用户输入的**字符串进行判断**：用户可能输入物理位置或逻辑位置，物理位置易于定位，而逻辑输入则比较繁琐。我们把逻辑输入分为三类，调用judgeStr（）函数进行判断

//对用户输入的字符串进行判断，判断顺序——课表考表判断——物理位置——逻辑位置

//我们提前已经知道了校区，传入的地图是另一个校区的地图

//若修改了pos就返回0，然后输出多点

//若查无此地返回-1

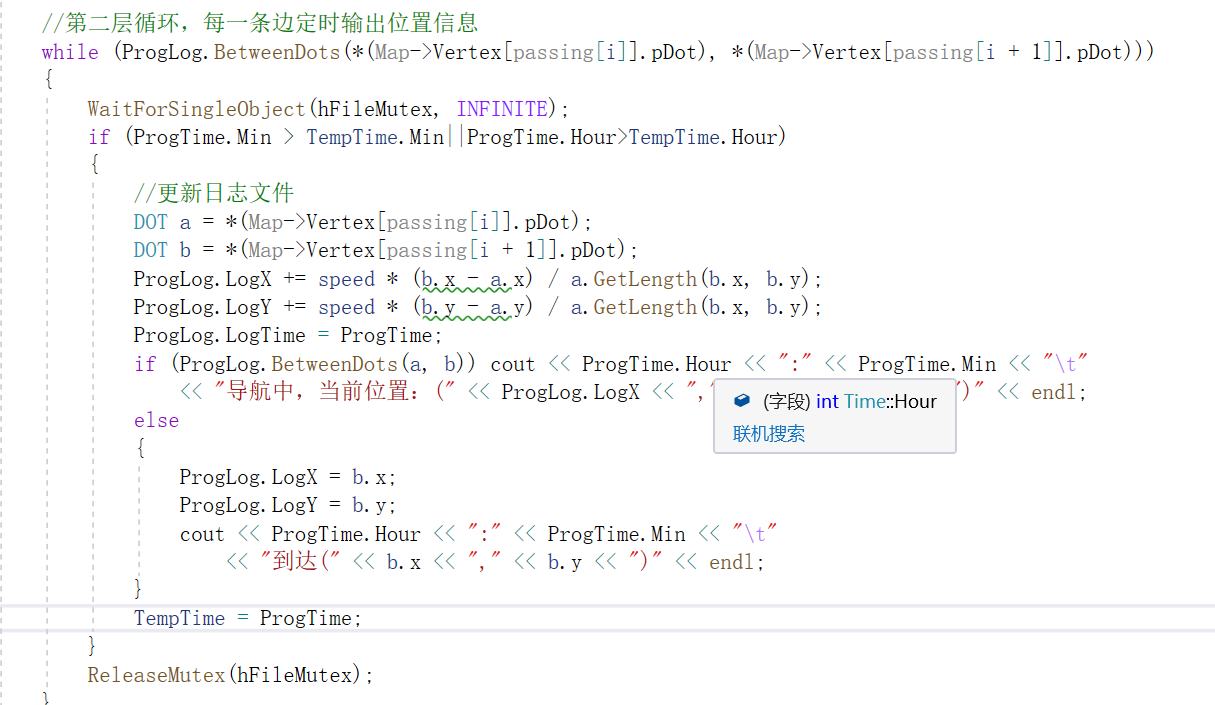
//如果课表上对应的校区与用户输入不匹配，更改school

int GRAPH::judgeStr(GRAPH\* map, string str, vector<int>& pos, classTime\* A, testTime\* B, int\* school)

当返回多点时，要输出所有符合输入的可能地点用confirm函数提供给用户让用户进行确认。

在确定了起点终点后，就可以依据校区进入不同导航模块。

同校区导航时，调用函数void guideInside(GRAPH\* map, int start, int end) 。内部的导航主体采用在GRAPH类内封装的**迪杰斯特拉算法**，复杂度为O(n^2)，返回用vector数组储存的路径。在使用迪杰斯特拉获得路径途径点的集合后，我们又自主写出了函数int guideOutput(GRAPH\* Map, vector<int> passing) 用于**输出导航信息**，此函数内部套用两层循环，一层遍历路径中各条边，另一层对每一条边**结合时间线程**定时输出位置信息，所以复杂度仍为O（n^2）。同时，此函数中简单根据坐标判断东南西北并输出，还兼顾更新日志文件的功能：

（输出中与时间线程结合的部分）

而在**跨校区导航**时，把导航全程拆分成两段校内导航和一段校区间导航，对于校区间导航，选择分别计算校车和公共汽车的时间进行对比，计算部分的算法仍然选择了可以直接调用的迪杰斯特拉，因此复杂度仍然是O（n^2）。

对于途径多点的导航情况，考虑到可行性与高效性，我们最终采用了**贪婪算法**，时间复杂度为O（n）。同时，考虑到多个校区的现实情况，程序同时实现了单校区途径多点导航和跨校区途径多点导航。

**5、日志文件处理模块**完成相应的日志文件写入和查询结果输出等功能。

定义了Log类型用来存放日志相关操作。

