



## 数据结构课程设计2022

## ——数据结构说明报告

小组成员： 韩旭、陈可儿、齐雨婷

学 院 ：计算机学院（国家示范性软件学院）

专 业 ： 计算机科学与技术

班 级 ： 2020211307

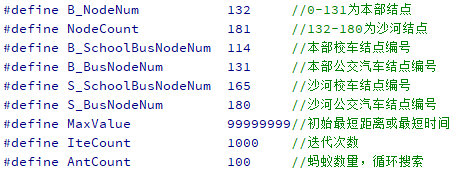
指导教师： 张海旸老师

**注：本报告将列举该系统中重点模块使用到的所有基本数据结构**

1. 校园导航

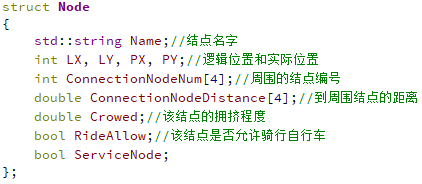
**navigation模块**

* 声明变量：

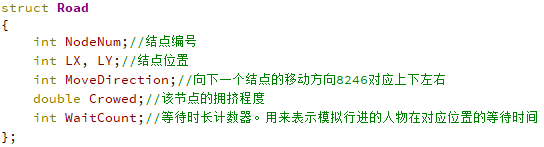


* 结构体变量

1. 结点



1. 道路



* 函数

|  |  |  |
| --- | --- | --- |
| 函数名 | 实现功能 | 分属类 |
| Ant() | 初始化蚂蚁信息，读取map.txt里的地图信息，并通过读取的信息，计算每个结点到周围方向结点的距离，并将对应的校车和公交车结点初始化，到周围结点距离为本部到沙河校区的路程。 | Ant |
| void ResetAnt() | 重置蚂蚁信息---除了地图结点信息，全部重置 | Ant |
| int FindMinTime(int startTime,int startLocation) | 如果坐校车或者公交车往返两个校区之间，根据车的班次、等待时间和运行时段，返回做车最短的时间,如果不是跨校区，则返回MaxValue | Ant |
| int SelectNextCityNum(bool ride) | 选择下一个前进的结点----先寻找到一条路径到达终点，然后再计算距离和时间的优先程度，向可以走的所有方向中随机选一个方向作为下一个前进结点 | Ant |
| void MoveAnt(bool ride, int nowTime) | 移动蚂蚁位置，并计算移动距离和花费时间 | Ant |
| void Search(int start, int finish, bool ride, int nowTime) | 根据布尔变量判断是否更改默认移动速度,如果是骑行速度改为2.4, | Ant |
| bool returnCross(int start, int end) | 传入起始和结束位置，如果不在一个校区，返回true，否则false | Navigation |
| void Search1(int start, int finish, bool bike, int nowTime) | 计算同一校区间两点的最短路径 | Navigation |
| void Search2(int start, int finish, bool bike, int nowTime) | 计算同一校区间两点的最短时间 | Navigation |
| void SearchTime(int start, int finish, bool bike, int nowTime) | 计算跨校区或者在同一校区的最短时间 | Navigation |

**qtsgs模块**

* 变量

、

* 函数

|  |  |
| --- | --- |
| 函数名 | 实现功能 |
| QtSGS(QWidget\* parent) | 将导航主界面和各项数据初始化 |
| void timerEvent(QTimerEvent\* e) | 模拟行进的底层代码逻辑，并且开始取决于Stimulation的值 |
| void paintEvent(QPaintEvent\*) | 将模拟行进路线在界面上显示出来 |
| bool SearchJudge() | 对于搜索选项选择的正确性判断 |
| void clearFinalRoad() | 清空导航路线，将道路数据初始化 |
| int CaculateDirection(int x1, int y1, int x2, int y2) | 根据8 2 4 6代表上下左右，计算行进方向 |
| bool checkBusLeaveTime(bool schoolType, bool busType) | 检测不同校区内校车和公交的发车时间，如果在发车时间内，返回true，否则false |
| void WriteData() | 将信息写入日志文件，从导航开始一直到导航结束 |
| void TimerFaster() | 加快虚拟时间 |
| void clickStartSearch() | 单击搜索按钮后触发的函数，会根据用户选择的策略进行判断 |
| void clickSwitchButton() | 点击切换地图时，是否在本部NowArea值转换，在本部为true，不在为false |
| void clickStopSearchButton() | 点击暂停按钮后的模拟行进停止，开始搜索按钮可点，暂停搜索按钮不可点，并且可切换地图 |
| void changeStartNodeNum() | 如果开始位置选择下拉框有任何操作。则更新对应的信息 |
| void changePassNodeNum() | 如果途经位置选择下拉框有任何操作。则更新对应的信息 |
| void changeEndNodeNum() | 如果结束位置选择下拉框有任何操作。则更新对应的信息 |
| void on\_pushButton\_clicked() | 点击进入主界面按钮 |
| void on\_changeTimeF\_2\_clicked() | 结束指定计时器ID计时 |
| void on\_changeTimeF\_3\_clicked() | 新的计时器，开始计时 |

1. 排序

* 结构体：

struct **TimeInfo**

{

int id;//用于存储记录标号

int week;//用于存储记录time字段的周次

int hour;//用于存储

};

* 类

class **TimeSort** : public QObject

{

Q\_OBJECT

public:

explicit **TimeSort**(QObject \*parent = nullptr);//构造函数，初始化TimeInfo t[200]

int **sort**(TimeInfo a[],int begin,int end);//单次排序

void **init**(TimeInfo t[], QList<EventInfo> lEvents);//根据记录time字段对数组进行初始化

void **QuickSortNoR**(TimeInfo\* arr, int begin, int end);//非递归快排

TimeInfo t[200];//存放每条记录的id,time字段的周次信息和时间信息

signals:

private:

eventSql \*m\_ptreventSql;//自定义的数据库类的指针

};

1. 解压缩

* 结构体：

struct **HuffmanTreeNode**

{

pair<unsigned char, int>Weight; //记录字符和其权重

string zippassword; //等待获取的哈夫曼编码

HuffmanTreeNode\* leftson; //左儿子指针

HuffmanTreeNode\* rightson; //右儿子指针

bool tag; //是否为叶子结点的标记，true为非叶子结点

};

//哈夫曼树结点

* 类：

class **FileCompress** : public QWidget

{

Q\_OBJECT

public:

explicit **FileCompress**(QWidget \*parent = nullptr);

void **zip**(QString path); //压缩函数

void **unzip**(QString path); //解压函数

protected:

void **DEL**(HuffmanTreeNode\*& root);

//在每次压缩后需要delete掉申请空间的哈夫曼树指针，采用递归遍历进行delete

void **BinaryString\_Init**();

//二进制字符串的初始化，即从传入文件得到的字符串按照哈夫曼编码获取二进制字符串

void **Weightmap\_Init**(QFile& in);

//参数是QFile型，是Qt的文件操作，该函数打开传入文件，遍历，形成一个字符串，并且初始化WeightMap

HuffmanTreeNode\* **MakehuffmanTreeNode**(int i);

//该函数的功能是创建单个哈夫曼树结点

void **HuffmanTreeVector\_Init**();

//将创建的哈夫曼结点放入Vector中

void **HuffmanTree\_Init**();

//建立哈夫曼树的函数

void **ZipPassword\_Init**(HuffmanTreeNode\* &root, string& password);

//对建成的哈夫曼树进行遍历，更新每个结点的zippassword,即哈夫曼编码

private:

map<unsigned char, int> weightmap; //权重映射

map<unsigned char, string>passwordmap; //哈夫曼编码映射

string source\_string; //读取传入文件形成的字符串

string binary\_string; //通过哈夫曼编码映射形成的二进制字符串

vector<HuffmanTreeNode\* > container; //哈夫曼结点储存的容器

map<string, int> zippassword; //解压时建立的解码映射

signals:

void **error**(); //信号

void **mysignal**(double per); //信号

public slots:

};

1. 查找

* 结构体：

typedef struct **point**

{

struct point \*pNext;

int nValue;

char Name[20];

}Point;

* 函数：

void **CreateHashTable**(Point \*arr[],const char \*src, QList<EventInfo> lEvents);//创建哈希表

void **HashSearch**(Point \*arr[],const char ch,int position[], string to\_search);

int\* **Search**(QList<EventInfo>, QString scan\_string);

int **KMP**(string, string);

int\* **getNext**(string p);

void **DestroyHashTable**(Point \*arr[]);

QString **getChineseSpell**(QString& src);

char **convert**(wchar\_t n);

bool **In**(wchar\_t start, wchar\_t end, wchar\_t code);

1. 时间冲突检测

* 结构体：

struct **EventInfo**

{

int id;

QString name;

QString teacher;

QString location;

QString time;

QString test\_time;

QString test\_loc;

QString group\_number;

};

struct **TimeInfo**

{

int id;//用于存储记录标号

int week;//用于存储记录time字段的周次

int hour;//用于存储

};

1. 模拟系统时间

* 类：（只展示了MainWindwo类中涉及到模拟时间的部分）

class **MainWindwo** : public QWidget

{

Q\_OBJECT

public:

explicit **MainWindwo**(QWidget \*parent = nullptr);

~***MainWindwo***();

void ***timerEvent***(QTimerEvent\*);

void ***paintEvent***(QPaintEvent\*);

private:

int day,hour, minute, second;//时分秒

int TimerID;//计时器ID

int myday,myhour,myminute;//设置的闹钟时间

};

1. 闹钟

* 类：

用户自设闹钟：

class **dlg\_setalarm** : public QDialog

{

Q\_OBJECT

public:

explicit **dlg\_setalarm**(QWidget \*parent = nullptr);

~***dlg\_setalarm***();

int setday;//获取用户输入的week

int sethour;//获取用户输入的hour

int setminute;//获取用户输入的时间

private slots:

void **on\_Button\_accepted**();

void **on\_Button\_rejected**();

private:

Ui::dlg\_setalarm \*ui;

};

1. 数据库相关

* 结构体：

事件（课程及活动）信息

struct **EventInfo**

{

int id;

QString name;

QString teacher;

QString location;

QString time;

QString test\_time;

QString test\_loc;

QString group\_number;

};

学生信息

struct **UserInfo**

{

QString username;

QString password;

QString aut;

};

* 类：

class **eventSql** : public QObject

{

Q\_OBJECT

public:

static eventSql \*ptreventSql;

static eventSql \***getinstance**()

{

if(nullptr == ptreventSql )

ptreventSql = new eventSql ;

return ptreventSql;

}

explicit **eventSql**(QObject \*parent = nullptr);

//初始化

void **init**();

//获取事件数量

quint32 **getEveCnt**();

//获取事件数据

QList<EventInfo> **getPageEvent**(quint32,quint32);

//更新表格

void **updateTable**(EventInfo);

//增加日程信息

bool **addEve**(EventInfo);

//管理员修改课程信息

bool **updateEveInfo**(EventInfo);

signals:

private:

QSqlDatabase m\_db;

};