开发者文档

activity:

Activity(string name, int start_week, int end_week, int day, int start_period, int end_period, string location, int activity_type); //根据活动信息构造课外活动

string getName() const; //获取活动名字

int getType() const;//获取活动类型

activityschedule:

void read_activities(string filename, int type);//从文件里读取活动并保存

ActivitySchedule(string personal_filename, string group_filename);//从文件里初始化活动表类

bool checkActivityConflict(int type, int start_week, int end_week, int day, int start_period, int end_period); // 检查时间冲突的辅助函数

void add_activity(Activity activity, string filename); // 添加活动

string getActivity(int week, int day, int period);//获取某段时间内存在的活动

QString find_activity(const string &name); // 查找活动

void delete_activity(string name, string personal_filename, int startWeek, int endWeek, int day, int startPeriod, int endPeriod); // 删除活动

int checkType(const string name); // 查看活动类型

void setSchedule(Schedule *sched);//设置课程表

void setTemporarySchedule(TemporarySchedule *tempSchedule);//设置临时事务表

activitywindow:

void onFindActivityClicked();//查询活动按钮的点击操作

void onAddActivityClicked();//添加活动按钮的点击操作

void onActivityCellClicked(int row, int col);//用户点击表格内容时调用

void onSortActivitiesClicked();//对课外活动排序

void createActivityTable();//初始化课外活动表

void updateActivityTable();//更新课外活动表

adminwindow:

void showAllStudents();//管理员显示学生信息

void addStudent();//添加学生

void deleteStudent();//删除学生

void modifyStudent();//修改学生信息

alarmwindow:

void addClicked();//添加提醒按钮点击操作

void delClicked();//删除提醒按钮点击操作

void changeClicked();//更改提醒按钮点击操作

void setAlarmlist();//初始化提醒表

CLOCK_TIME

std::vector<alarm> alarm_list;//当前用户所有的提醒
std::vector<alarm> now_alarm_list;//当前时间点该进行的提醒
void CONTINUE();//继续计时
void PAUSE();//暂停计时
int getYear();//获取当前时间运行了多少年
int getMonth();//获取当前时间运行了多少月
int getDay();//获取当前时间运行了多少天
int getHour();//获取今天的小时
int getWeekDay();//获取今天周几
int getWeekNum();//获取今天是第几周
void alm_for_tonight();//每晚八点提醒第二天的活动
void alm_for_hour();//事情发生前一小时提醒一次
void alm_for_compaign();//每天早上八点提醒今天的活动
void addAlarm(string name,int weekNum,int week,int hour,string place_link);//想提醒表添加提醒
void delAlarm(int x);//从提醒表中删除提醒
void changeAlarm(int x, string name,int weekNum,int week,int hour,string place_link);//更改提醒表的某个提醒
void counter();//并发计时函数

graph:

void initVNode();// 初始化顶点表
unsigned int BKDRHash(char* str); //将地点名称的字符串映射到一个非负整数作为数组的下标值
void addEdge(string src, string dest, int flag); // 函数用来向图中添加一条边
unsigned int getIndex(string name);//通过地名获取数组下标
void printGraph();// 函数用来打印图的邻接表表示
void dijkstra(string src, string dest, int mode); // 用 dijkstra 算法计算结果
void getPath(unsigned int s, int* prev, int* transportation, int* temp_dist, int mode, double* congestion); // 找到最短路径
void printPath(unsigned int s, unsigned int d, int mode); // 输出最短路径
// 禁忌搜索算法
// 禁忌（Tabu Search）算法是一种亚启发式(meta-heuristic)随机搜索算法
// 它从一个初始可行解出发，选择一系列的特定搜索方向（移动）作为试探，选择实现让特定的目标函数值变化最多的移动
// 为了避免陷入局部最优解，TS 搜索中采用了一种灵活的“记忆”技术
// 对已经进行的优化过程进行记录和选择，指导下一步的搜索方向，这就是 Tabu 表的建立
int calcPathLen(const vector<int>& path, int mode); // 计算路径长度

void swapVal(int& a, int& b); // 交换两个位置的值
vector<int> generateCandidate(const vector<int>& cur_path); // 生成候选解
bool isInTabuList(const vector<int>& path); // 判断路径是否在禁忌表中
oid updateTabuList(const vector<int>& cur_path, const vector<int>& new_path); // 更新禁忌表和禁忌期
vector<int> tabuSearch(string src, vector<string> dst, int mode); // 禁忌搜索算法
void dfs(vector<int>& nums, vector<int>& visited, vector<int>& permutation, vector<int>& best_path, int& lowest_cost, int mode); // 利用 dfs 算法求解 TSP 问题
void show_dijk(string src, string dest, int mode); // 显示 dijkstra 导航结果
void show_dfs(string src, vector<string> dst, int mode);//显示 dfs 的导航结果
void show_tabu(string src, vector<string> dst, int mode);//显示 tabu 算法的导航结果

navigation:

void DrawLine(QLine line, double time, int mode, double congestion); // 绘制导航路线
void show_congest(double congest); // 显示拥塞度
void delete_lines();// 清空画线
void paintEvent(QPaintEvent*);// 绘制导航
void mousePressEvent(QMouseEvent* e); // 点击地图获取地图建筑物的坐标

log:

void log(std::string str)//向日志文件输出日志信息

loginwindow:

void onLoginButtonClicked();////登陆按钮点击操作
void authenticate(const QString &username, const QString &password);//验证用户的账号是否有效以及判断用户是否为管理员

person:

void readUserAccounts();//读取用户账号信息
void readCourses();//读取用户课程信息
void readActivities();//读取用户活动信息
void readTempEvents();//读取用户临时事务信息

schedule:

string getcourses(int week, int day, int period); // 获取课程
string getexams(int week, int day, int period); // 获取考试信息
QString FindCourse(const string &course); // 查找课程，course 为课程名字
void AddCourse(int startWeek, int endWeek, int day, int startPeriod, int endPeriod, string course, string teacherName, string classroom, int examTime, int examDay, int examStartPeriod, int examEndPeriod, string examRoom); // 添加课程
void DeleteCourse(int startWeek, int endWeek, int day, int startPeriod, int endPeriod); // 删除课程，startWeek 为开始周数，endWeek 为结束周数，day 为星期几，startPeriod 为开始节数，

endPeriod 为结束节数

bool checkCourseConflict(int start_week, int end_week, int day, int start_period, int end_period); // 检查课程是不是和其他课程冲突

bool checkOtherConflict(int start_week, int end_week, int day, int start_period, int end_period); // 检查课程是不是和活动，临时事物冲突

schedulewindow:

void onFindCourseClicked();//查找课程按钮点击操作
void onAddCourseClicked();//添加课程按钮点击操作
void onCourseCellClicked(int row, int col);//表格内容点击操作
void createCourseTable();//初始化课程表
void updateCourseTable();//更新课程表

sortedtempdialog:

void setupTable(const std::vector<TemporaryThing> &sortedThings);//临时事务窗口排序

sortresultdialog:

void setupTable();// 用于设置表格内容的函数
void onSortOptionChanged(int index); // 当下拉框选项改变时触发的槽函数

temporaryschedule:

bool checkTempScheduleConflict(int week, int day, int hour);//检查临时事务的冲突 bool checkTempConflict(int week, int day, int hour);
string getTemp(int week, int day, int hour);//获取特定时间的临时事务的名字
void add_thing(TemporaryThing thing, string filename); // 添加临时事物
QString find_thing(const string &name); // 查找临时事务
void delete_thing(string name, string filename); // 删除临时事务
void sort_things();// 排序临时事务

temporarythings:

bool TemporaryThing::isOverlapping(const TemporaryThing& other) const// 判断两个临时事物是否重叠
bool TemporaryThing::isValid() const // 判断临时事物的时间是否在有效范围内

tempwindow:

oid onFindTempClicked();//查找临时事务点击操作
void onAddTempClicked();//添加临时事务点击操作
void onTempCellClicked(int row, int col);//点击表格内容
void sortTempByTime();//按时间顺序排序临时事务
void createTempTable();//初始化临时事务
void updateTempTable();//更新临时事务

版本更新记录

V0.1_20230402

完成了用户登录模块，可进行登录、密码修改等操作。

初步实现了管理员的功能，可添加学生的学号和姓名。

初步完成了课程的添加删除更改查找功能。

V0.2_20230409

完善了课程的增删改查操作。

初步构造了导航模块的地图。

基本实现了提醒表的每日以及一小时提醒。

V0.3_20230416

完成了课外活动模块的添加删除及查找操作。

初步完成了冲突检测功能。

初步实现了寻找两点间最短导航路径功能。

初步实现了单起点多中间点并回到起点的最短路径搜索。

完善了提醒序列的增删改操作。

V0.4_20230423

实现了课外活动按时间排序的功能。

初步完成提醒表的信息结构。

完善了寻找两点间最短导航路径功能。

完善了单起点多中间点并回到起点的最短路径搜索。

V0.5_20230430

完善了添加课外活动时的冲突检测功能。

初步构建了临时事务。

设计了提醒与模拟时钟的图形界面。

优化了寻找两点间最短导航路径功能。

优化了单起点多中间点并回到起点的最短路径搜索算法。

V0.6_20230507

完善了课程和课外活动模块。

初步实现了提醒与模拟时钟的图形界面。

优化了导航的路径输出。

初步设计了各模块的图形界面。

V0.7_20230514

完善了导航模块的 dijkstra 算法。

初步构建了地图的图形界面。

完善了提醒与模拟时钟的图形界面。

实现了临时事务与课程和活动之间的冲突检测功能。

V0.8_20230521

优化了闹钟与时钟的显示界面。

实现了课程、活动与临时事务模块的图形界面。

实现了地图导航的图形界面。

V0.9_20230528

完善了导航模块的图形界面。

完成了日志模块的构建。

完成了各模块之间的对接联合。

完善了冲突检测的图形界面。