数据结构课程设计实验报告

竹倩叶 2017211312班 2017211444

付彬 2017211309班 2017211357

一、任务描述

功能描述：城市（总数不少于10个）之间有三种交通工具（汽车、火车和飞机）相连；某旅客于某一时刻向系统提出旅行要求（包括：起点、终点、途经某些城市和旅行策略），系统根据该旅客的要求为其设计一条旅行线路并输出；系统能查询当前时刻旅客所处的地点和状态（停留城市/所在交通工具）；

详细需求及分析：

1、旅行策略有：

（1）最少费用策略：无时间限制，费用最少即可，

（2）最少时间策略：无费用限制，时间最少即可，

（3）限时最少费用策略：在规定的时间内所需费用最省；

2、建立汽车、火车和飞机的时刻表（航班表），有沿途到站及票价信息，使用txt格式文档存储；

3、旅行模拟查询系统以时间为轴向前推移，每10秒左右向前推进1个小时(非查询状态的请求不计时)；系统时间精确到小时；实现模拟旅客旅行的过程；

4、不考虑城市内换乘交通工具所需时间，旅客可以选择在城市内停留一段时间；

5、建立日志文件，对旅客状态变化和键入等信息进行记录；

6、某旅客在旅行途中可更改旅行计划，系统应做相应的操作；

7、用图形绘制地图，并在地图上反映出旅客的旅行过程；

二、方案设计说明

1、软件开发环境：

Visual Studio 2017、QT

2、总体结构：

3、模块划分：

三、数据结构与数据字典

1、结构体：

（1）/\*航班表结构\*/

typedef struct trans\_t {

int src, dest;//起点和终点

Vehicle transport;//车型//枚举类型

char name[10];//交通工具名称

int number;//车次

int time\_departure;//发车时间 //系统时间精确到小时

int time\_consumed;//耗时

int cost;//钱

struct trans\_t \*nextPtr;

}trans\_t\_Node, \*Ptr\_trans\_t\_Node;

（2）边 结构

typedef struct Edge {

struct trans\_t \*p\_TransTable;//此边上的航班表

int weight;//边的权重//是可以唯一确定的

int num\_OfTheEgde;//权重最小的边的编号

int distance;//城市之间的距离，0代表本市到本市，-1代表不连通

}EDGE, \*P\_EDGE;

（3）图（城市）-邻接矩阵 结构

typedef struct graph {

EDGE \*\*pp\_G;//指向邻接矩阵(二维动态数组）的指针//大小为Graph\_size\*Graph\_size

int Graph\_size;//城市数量

char \*\*City\_Name;//城市名称

}GRAPH;//定义了结构体方便后期扩展

（4）栈 结构

typedef struct {

int \*base, \*top, stack\_size;//栈底、栈顶、栈大小

}SqStack;

（5）系统时间 结构

typedef struct SYSTEM\_TIME {

int year, month, date, hour;//年、月、日、小时

}SYSTEM\_TIME;

（6）旅客旅行状态 结构

typedef struct User\_Status

{

SYSTEM\_TIME time;//当前时刻

Location loca;//乘客旅行状态

int src, dest;//起点，终点；若乘客在途中，代表交通工具的起始点；若乘客停留在某城市，src=dest；

char name[10];//交通工具名称

}User\_Status;

（7）乘客 结构 链表形式存储

typedef struct passenger {

char ID[20];//旅客名称

int src, dest;//起点终点

int strategy;//旅行策略//时间or费用最少or限制时间最少费用//STRA\_minCOST, STRA\_minTIME, STRA\_limTIME\_minCOST

int num\_passby;//途经节点数量

int pass\_by[2][MAX\_NODE\_NUM];//row0要求途经的城市;row1在该地停留时间

int Time\_Limited;//若限制时间最少费用，还有一项限制的时间

SYSTEM\_TIME start\_time;//旅行者开始旅行的时间

User\_Status status;//旅客旅行状态

struct passenger \*next\_passenger;//指向下一个旅客

}PASSENGER;

（8）旅客旅行路线的结点 结构

typedef struct pathnode

{

int src;//一段路径的起点

int dest;//一段路径的终点

int number;//旅行者搭乘的车次

SYSTEM\_TIME start\_time;//开始时间

int time;//耗费时间

int cost;//花费

struct pathnode \*next;

}PathNode, \*PATH;

2、全局变量：

bool Quit = false;//退出系统

SYSTEM\_TIME System\_Time;//系统时间

bool inputing = false;//暂停系统时间标志

GRAPH city\_graph = { NULL, 0, NULL };//地图

PASSENGER \*Passengers = NULL, \*Passengers\_tailPtr = NULL, \*User;//User当前系统使用者

FILE \*fptr\_input;//日志文件，记录用户输入信息

int Travelstate[10] = { 0 };//假定最多10个旅客，缓存每个旅客的当前旅行到的第几个城市

3、枚举类型常量：

enum Vehicle { CAR, TRAIN, AIRPLANE };//交通工具

enum Status { ERROR, OK, STACK\_FULL, STACK\_EMPTY, UNABLE };//程序状态

enum Travel\_Strategy { STRA\_minCOST, STRA\_minTIME, STRA\_limTIME\_minCOST };//旅行策略

enum Location { IN\_CAR, IN\_TRAIN, IN\_AIRPLANE, STAY\_IN\_CITY, ARRIVE };//旅客状态

四、主要模块的设计说明

**1、时间线程**

每10秒时间加1，更新系统中的全局变量System\_Time，同时刷新用户的旅行状态，会为所有旅客调用刷新状态的函数。

如果用户正在注册、登录，则会暂停时间。

在选择退出系统后，写文档，释放内存，关闭程序。

程序如下：

unsigned \_\_stdcall time(void\* pArguments)

{

while (Quit == false)

{

if (!inputing)

{

Sleep(10000);

System\_Time.hour++;

if (System\_Time.hour == 24)

{

System\_Time.date++;

System\_Time.hour = 0;

}

if (System\_Time.date == 31)//让我们假设每个月都30天

{

System\_Time.month++;

System\_Time.date = 1;

}

if (System\_Time.month == 13)

{

System\_Time.year++;

System\_Time.month = 1;

}

//对所有旅客刷新其旅行状态

PASSENGER \*temp = Passengers;

int touristnum = 0;

while (temp != NULL)

{

if (temp->status.loca != ARRIVE && Travelstate[touristnum] != 0)

Refresh(temp, touristnum);

touristnum++;

temp = temp->next\_passenger;

}

}

else if (inputing)

{

Sleep(500);//休眠0.5s

}

}

/\*写需要保存的系统状态\*/

Write\_system\_file();

//释放动态申请的内存

Freememory();

\_endthreadex(0);

return 0;

}

**2、刷新旅客状态**

从路线文档中读取出旅客当前旅行计划，

**3、用户操作线程**

**4、注册**

**5、登录**

**6、查询旅行状态**

**7、开始旅程**

**8、修改旅行计划**