Project 4

中山大学校园导游系统

班级： 软工5班

盘学之 16340177

谭发豪 16340195

苏泽华 16340195

**【题目要求】**

1. 从中山大学东校区的平面图中选取有代表性景点（10-15个），抽象成一个无向带权图。以图中顶点表示校内各景点，存放景点名称、代号，简介等信息；以边表示路径，存放路径长度等信息；
2. 为来访客人提供图中任意景点相关信息的查询；
3. 为来访客人提供图中任意景点的问路查询，即查询任意两个景点之间的一条最短的简单路径；
4. 区分汽车线路与步行线路；

**【数据结构与算法】**

数据结构：

typedef struct vertex { //构造地图节点的结构

int number; //节点的编号

string name; //节点的名字

string info; //节点的相关信息

bool is\_scene; //是否景点

int x;

int y; //点的坐标

vertex(int n = 0, string str = "", string i = "", bool s = true) { //初始化节点

number = n;

name = str;

info = i;

is\_scene = s;

}

} vertex;

typedef struct edge { //构造道路的结构

int weight; //权重(长度)

bool is\_pavement; //是否人行道

bool is\_driveway; //是否车道

edge(int w = 0, bool p = false, bool d = false) { // 初始化道路

weight = w;

is\_pavement = p;

is\_driveway = d;

}

} edge;

class map { //构造地图的类

public:

map(): vertex\_num(0), edge\_num(0) { //构造函数

for(int i = 0; i < MAX\_VERTEX; i++) {

for(int j = 0; j < MAX\_VERTEX; j++) {

adjacency\_matrix[i][j] = nullptr;

}

vertex\_list[i] = nullptr;

}

}

~map() { //析构函数

for(int i = 0; i < vertex\_num; i++) {

for(int j = i; j < vertex\_num; j++) {

if(adjacency\_matrix[i][j] != nullptr) {

delete adjacency\_matrix[i][j];

delete adjacency\_matrix[j][i];

}

}

if(vertex\_list[i] != nullptr)

delete vertex\_list[i];

}

}

bool ReadFromFile(string url) { //文件读取函数，从文件中读入地图信息

// 打开数据文档

ifstream file;

file.open(url);

//打开文档失败，返回false

if(!file)

return false;

//处理数据并存储，最终返回true

string temp = "";

int pos1 = 0;

int pos2 = 0;

file >> temp >> vertex\_num >> temp;

for(int i = 0; i < vertex\_num; i++) {

vertex\_list[i] = new vertex;

file >> vertex\_list[i]->number;

getline(file, temp);

getline(file, vertex\_list[i]->name);

getline(file, vertex\_list[i]->info);

file >> vertex\_list[i]->is\_scene;

}

file >> temp >> edge\_num >> temp;

for(int i = 0; i < edge\_num; i++) {

file >> pos1 >> pos2;

adjacency\_matrix[pos1][pos2] = new edge;

file >> adjacency\_matrix[pos1][pos2]->weight

>> adjacency\_matrix[pos1][pos2]->is\_pavement

>> adjacency\_matrix[pos1][pos2]->is\_driveway;

adjacency\_matrix[pos2][pos1] = adjacency\_matrix[pos1][pos2];

}

file.close();

return true;

}

bool WriteToFile(string url) { //

// 打开数据文档

ofstream file;

file.open(url);

// 写入数据

file << "vertex\_num: " << vertex\_num << endl << endl

<< "vertex\_list(number,name,info,scene?):" << endl << endl;

for(int i = 0; i < vertex\_num; i++) {

file << vertex\_list[i]->number << endl

<< vertex\_list[i]->name << endl

<< vertex\_list[i]->info << endl

<< vertex\_list[i]->is\_scene << endl << endl;

}

file << "edge\_num: " << edge\_num << endl << endl

<< "edge\_list(vertex1,vertex2,weight,pavement?driveway?):" << endl;

for(int i = 0; i < vertex\_num; i++) {

for(int j = i; j < vertex\_num; j++) {

if(adjacency\_matrix[i][j] != nullptr) {

edge\* pointer = adjacency\_matrix[i][j];

file << i <<" "<< j <<" "<< pointer->weight <<" "

<< pointer->is\_pavement <<" "<< pointer->is\_driveway

<< endl;

}

}

}

file.close();

return true;

}

// 测试用的函数

// 用来初始化数据

void init() {

//init

vertex\_num = 5;

edge\_num = 6;

for(int i = 0; i < 5; i++) {

vertex\_list[i] = new vertex;

vertex\_list[i]->number = i;

}

vertex\_list[0]->name = "canteen 1";

vertex\_list[0]->info = "go to eat dinner!!";

vertex\_list[1]->name = "canteen 2";

vertex\_list[1]->info = "go to eat lunch!!";

vertex\_list[2]->name = "library";

vertex\_list[2]->info = "for books";

vertex\_list[3]->name = "doom";

vertex\_list[3]->info = "nothing";

vertex\_list[4]->name = "home";

vertex\_list[4]->info = "WTF?";

adjacency\_matrix[0][1] = new edge(23,0,1);

adjacency\_matrix[1][0] = adjacency\_matrix[0][1];

adjacency\_matrix[0][3] = new edge(5,1,1);

adjacency\_matrix[3][0] = adjacency\_matrix[0][3];

adjacency\_matrix[4][1] = new edge(24);

adjacency\_matrix[1][4] = adjacency\_matrix[4][1];

adjacency\_matrix[2][3] = new edge(11,1,0);

adjacency\_matrix[3][2] = adjacency\_matrix[2][3];

adjacency\_matrix[2][4] = new edge(25);

adjacency\_matrix[4][2] = adjacency\_matrix[2][4];

adjacency\_matrix[3][4] = new edge(23,1,1);

adjacency\_matrix[4][3] = adjacency\_matrix[3][4];

return;

}

// 用以输出

void test\_print() {

cout << "vertex\_num: " << vertex\_num << endl

<< "edge\_num: " << edge\_num << endl << endl;

for(int i = 0; i < vertex\_num; i++) {

cout << vertex\_list[i]->number << " " << vertex\_list[i]->name<<endl;

}

cout << endl;

for(int i = 0; i < vertex\_num; i++) {

for(int j = i; j < vertex\_num; j++) {

if(adjacency\_matrix[i][j] != nullptr) {

cout << i <<" "<< j <<" ";

cout << adjacency\_matrix[i][j]->weight << endl;

}

}

}

}

int getX(int num)

{

return vertex\_list[num]->x;

}

int getY(int num)

{

return vertex\_list[num]->y;

}

string getInfo(int num)

{

return vertex\_list[num]->info;

}

vertex\* getP(int num)

{

return vertex\_list[num];

}

//得到点的内部信息

int get\_number(const string& name) {

for(int i = 0; i < vertex\_num; i++) {

if(vertex\_list[i]->name == name) {

return i;

}

}

return -1;

}

// 传入点的名字，返回点的编号；若点不存在，返回-1

private:

//const int MAX\_VERTEX = 50; //最大结点数量

int vertex\_num; //结点数量

int edge\_num; //边数量

// 邻接矩阵，两点相通则有对应位置的edge指针有值，其余全为 nullptr

edge\* adjacency\_matrix[MAX\_VERTEX][MAX\_VERTEX];

// 结点数组，有新的点则new一个新值

vertex\* vertex\_list[MAX\_VERTEX];

};

算法:

**【测试数据、结果及分析】**

（所输入的数据及相应的运行结果，运行结果要有提示信息，运行结果采用截图方式给出。）

**【分工、贡献%、自我评分】**

分工：

盘学之：图像处理、调试、优化代码

谭发豪：底层开发、调试、优化代码

苏泽华：UI设计、撰写报告、优化代码

贡献：

盘学之：33.3%

谭发豪：33.3%

苏泽华：33.3%

自我评分：

盘学之：95

谭发豪：80

苏泽华：85

**【项目总结】**

（收获、体会，若实验课上未完成调试，要认真找出错误并分析原因等。）

本次项目的自由度比前两次都大，这样给了我们更多的发挥空间，但同时也给我们更大的开发压力。在这一次项目中，我们更真切地感受到了“软件工程”的“工程”体现，即团队成员各司其职又能无缝对接的丝滑般体验，觉得颇有成就感。不过这成就感背后少不了的还是血与泪。当我们提出要用其他方式、而不仅仅是命令行来表现族谱的时候，就无疑是给自己挖了一个大坑，需要自己不断学习新的知识和方法来“填坑”。

遗憾还是有的，就是我们或许应该用QT来写一个用户界面，不过因为时间关系我们没有做出来，希望下一个项目可以把这个坑也填一下吧。除此以外，这一次开发过程也是走了不少弯路，比如做过一些没有用的功能，浪费了不少时间，还有一开始变量名设计得不好，影响了可读性……这些都是我们在下一次项目开工前要做好防范措施的，否则只会再度浪费时间。

当然，除了上面的遗憾，我们这个家谱管理系统还是有一些不完善的地方，比如：在添加子辈的时候，暂时没有设计子代年龄不能大于父辈的限制；没有设计如果没有伴侣就不能添加子代的限制；输入日期的格式错误时没有报错，只是将日期改为：0000-00-00；暂时不支持名字含空格；不支持中文名……

总而言之，对于本次项目，我们总体还是比较满意的，非常期待下一次作业。

**【程序清单**】

main.cpp、Date.hpp、Date.cpp、struct.hpp、struct.cpp、xml.css

由于word的缩进与高亮不适用于代码展示，源代码详请见附件