图与景区信息管理系统实践

刘　　春

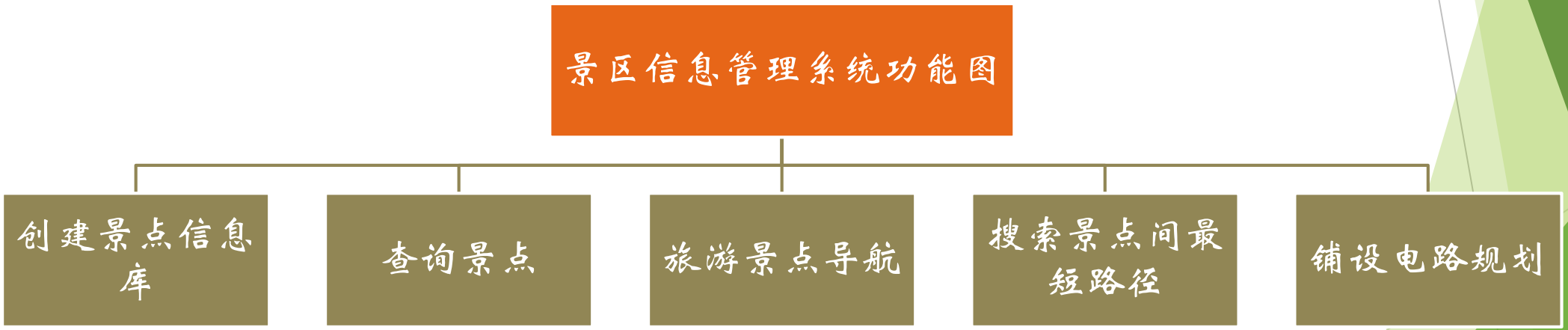
2019-5

实验目标

- 掌握图的定义和图的存储结构
- 掌握图的创建方法和图的应用
- 采用迭代开发思路实现“景区信息管理系统”

实验任务

- 现有一个景区，景区里面有若干个景点，开发景区信息管理系统，对景区的信息进行管理。



难点内容 → 图的存储结构

图的结构比较复杂，任意两个顶点之间都有可能存在联系，因此无法以数据元素在存储区中的物理位置来表示元素之间的关系，即图没有顺序映像的存储结构。

图的存储结构除了要存储图中各个顶点本身的信息外，同时，还要存储顶点与顶点之间的所有关系（边的信息）。

常用的图的存储结构有**邻接矩阵**和邻接表。

重点内容 ⟶ Tourism类的构建

```
1   #pragma once
2   #include "Graph.h"
3   class Tourism
4   {
5   private:
6       Graph graph;
7   public:
8       Tourism();
9       ~Tourism();
10      //建立景点信息库
11      void CreateGraph();
12      void ShowGraph();
13      //查询景点信息
14      void GetSpotInfo();
15      //景区导航
16      void TravelPath(int type);
17      //找最短路径
18      void FindShortPath(void);
19      //电路规划
20      void DesignPath(void);
21  };
22
```

创建景点信息库

查询景点

旅游景点导航

搜索景点间最短路径

铺设电路规划

难点内容 → Graph类的构建

```
//顶点信息
struct Vex
{
    int num;
    char name[StrMAX];
    char desc[StrMAX];
};

//边的信息
struct Edge
{
    int vex1;
    int vex2;
    int weight;
};

//深度优先遍历的路径
typedef struct Path {
    int Vexs[numMAX];
    Path *next;
} *PathList;
```

```
class Graph
{
private:
    int AdjMatrix[numMAX][numMAX];    //邻接矩阵
    Vex Vexs[numMAX];                 //点的集合
    int VexNum;                       //点的个数
public:
    Graph();
    ~Graph();
    bool InsertVex(Vex svex);
    Vex GetVex(int index);
    void SetVexNum(int);
    int GetVexNum(void);
    bool InsertEdge(Edge sedge);
    Edge GetEdge(int vex1, int vex2);
    int FindAdjSpots(int index, Edge aEdge[]);
    int DFSTraverse(int nVex, PathList &pList);
    int DFSTraverse2(int nVex, PathList &pList);
    int FindShortPath(int nVexStart, int nVexEnd, int aPath[], int &number);
    int FindMinTree(Edge aPath[]);
private:
    void DFS(int nVex, bool bVisted[], int &nIndex, PathList &pList);
    void DFS2(int nVex, bool bVisted[], int &nIndex, PathList &pList);
};
```

难点内容 → 创建景点信息库

```
//建立景点信息库
void Tourism::CreateGraph()
{
    //1.init graph
    ifstream File;
    char buffer[100];
    int vexnum;

    //2.set the graph vex

    File.open("Vex.txt", ios::in);
    if (File) { ... }
    else { ... }
    File.close();

    //2.set the graph edge
    File.open("Edge.txt", ios::in);
    if (File) { ... }
    else { ... }
    File.close();
}
```

难点内容 → 查询景点

```
//查询景点信息
void Tourism::GetSpotInfo()
{
    cout << endl;   cout << endl;
    int i;
    cout << "请输入想要查询的顶点编号：";
    cin >> i;
    Vex v = graph.GetVex(i);
    cout << v.name << endl;
    cout << v.desc << endl;
    Edge aEdge[numMAX];
    int num = graph.FindAdjSpots(i, aEdge);
    cout << "----周边景区----" << endl;
    for (i = 0; i < num; i++)
    {
        Edge e = aEdge[i];
        Vex start = graph.GetVex(e.vex1);
        Vex end = graph.GetVex(e.vex2);
        cout << start.name << "->" << end.name << "    " << e.weight << "米" << endl;
    }
}
```

难点内容 → 景点导航

```
//景区导航
void Tourism::TravelPath(int type)
{
    cout << endl;    cout << endl;
    int i;
    cout << "====旅游景点导航====" << endl;
    for (int i = 0; i < graph.GetVexNum(); i++)
    {
        Vex v = graph.GetVex(i);
        cout << v.num << "-" << v.name << "--" << v.desc << endl;
    }
    cout << "请输入起始点编号:";
    cin >> i;

    if (i<0 || i>graph.GetVexNum())
    {
        cout << "输入顶点有错！\n";
        return;
    }
```

```
PathList pathList=(PathList)malloc(sizeof(Path));
pathList->next = NULL;
PathList pathHead = pathList;

//调用深度优先导航路线
int num;
if(type==1)
    num = graph.DFSTraverse(i, pathList);
else
    num= graph.DFSTraverse2(i, pathList);

//输出导航路线
cout << "导游路线：" << endl;
i = 1;
pathList = pathHead;
while (pathList){ ... }

free(pathList);
pathList = NULL;
}
```

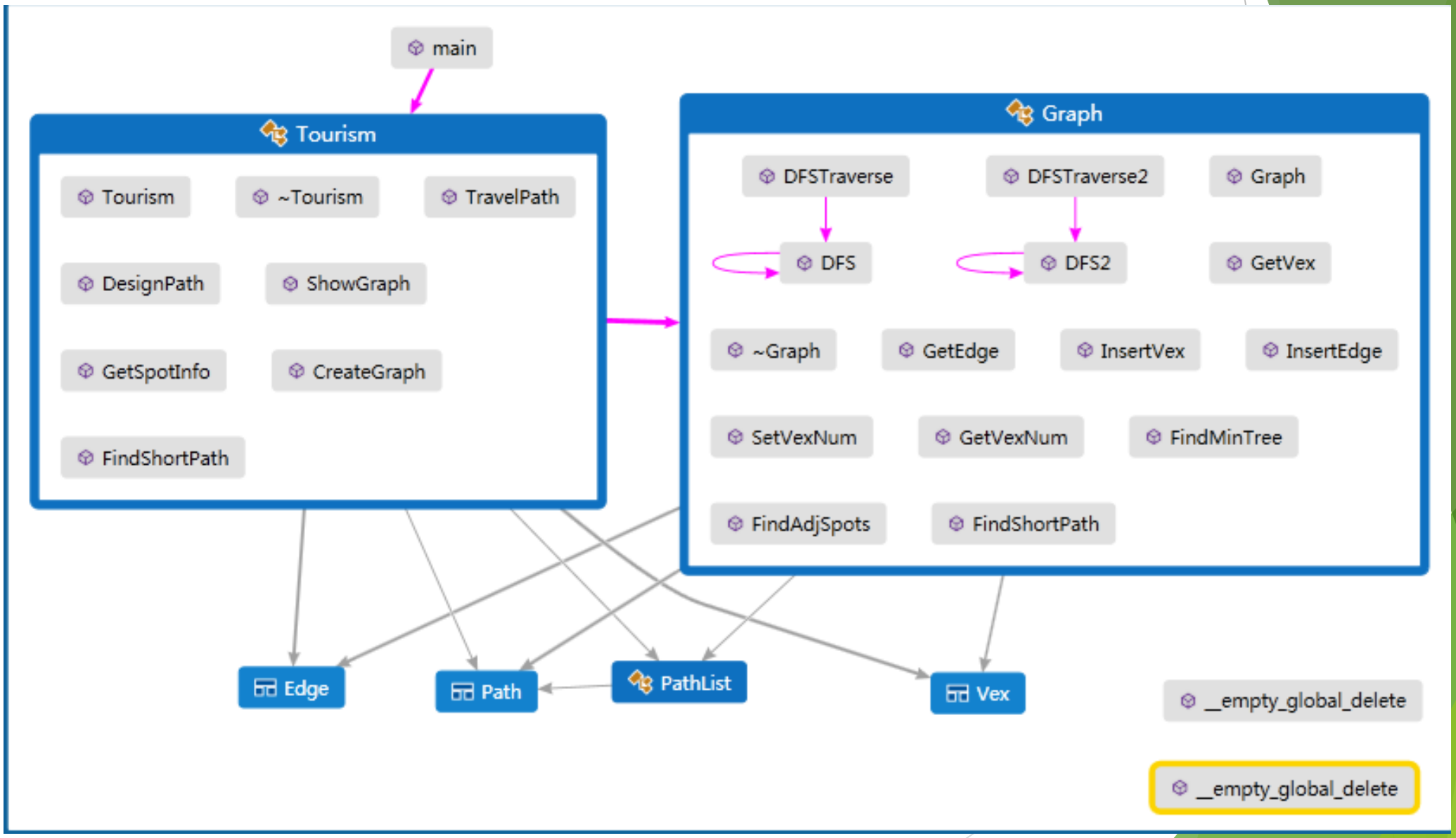
难点内容 → 最短路径

```
//找最短路径
void  Tourism::FindShortPath()
{
    cout << endl;    cout << endl;
    cout << "====搜索最短路径====" << endl;
    for (int i = 0; i < graph.GetVexNum(); i++)
    {
        Vex v = graph.GetVex(i);
        cout << v.num << "-" << v.name << "--" << v.desc << endl;
    }
    int startVex, endVex;
    cout << "请输入起始点编号:";
    cin >> startVex;
    cout << "请输入终止点编号:";
    cin >> endVex;

    int aPath[numMAX] = {MAX};
    int number = 0;
    int shortPath = graph.FindShortPath(startVex, endVex, aPath,number);

    //打印结果

    if (shortPath != MAX) { ... }
    else { ... }
}
```

难点内容 → 电路铺设

```
//电路规划
void Tourism::DesignPath(void)
{
    Edge aPath[numMAX];
    int sum = graph.FindMinTree(aPath);
    //输出结果
    cout << endl;   cout << endl;
    cout << "====铺设电路规划====" << endl;
    cout << "在以下两点间铺设电路" << endl;
    for (int i = 0; i < sum; i++)
    {
        cout << graph.GetVex(aPath[i].vex1).name << " - ";
        cout << graph.GetVex(aPath[i].vex2).name << "   ";
        cout << aPath[i].weight << "米" << endl;
    }
}
```

完整的代码结构图


main
Tourism
Tourism
~Tourism
TravelPath
DesignPath
ShowGraph
GetSpotInfo
CreateGraph
FindShortPath
Graph
DFSTraverse
DFSTraverse2
Graph
DFS
DFS2
GetVex
~Graph
GetEdge
InsertVex
InsertEdge
SetVexNum
GetVexNum
FindMinTree
FindAdjSpots
FindShortPath
Edge
Path
PathList
Vex
__empty_global_delete
__empty_global_delete