广州大学学生实验报告

开课实验室： 电子楼 416B　　2018 年 5 月 13 日

学院	计算机科学与网络工程学院	年级、专业、班	软件 171	姓名	谢金宏	学号	1706300001
实验课程名称	数据结构					成绩	
实验项目名称	实验三　图的遍历生成树					指导老师	杜娇

一、实验目的

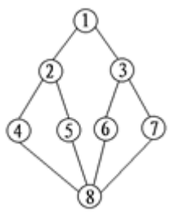
1. 熟悉图结构。
2. 掌握图结构上的各种操作。
3. 学会运用图结构求解问题。

二、使用仪器、器材

1. 使用 Windows 操作系统的微型计算机一台。
2. 符合 C++11 标准的 C++编程软件。

三、实验内容及原理

编写程序实现对下图的先深、先广遍历。



具体要求：
使用图的邻接矩阵表示法进行编程，实现如下基本接口：
FirstAdj(v)：找到编号为 v 的顶点的第一个邻接顶点。
NextAdj(v, w)：设 w 是 v 的邻接顶点，找到 v 的排在 w 后的下一个邻接顶点。
DepthFirstSearch(v) 对连通图从顶点 v 开始进行深度优先访问。
BroadFirstSearch(v) 对连通图从顶点 v 开始进行广度优先访问。

四、实验过程原始数据记录

实验使用的源代码如下：

Graph.cpp

图的定义

```
const int MAX_GRAPH_SIZE = 16;

int VertexCount;                                    // 图中顶点
bool IsVisited[MAX_GRAPH_SIZE];                     // 顶点的访问标志，Visited[i]
表示在一次遍历中顶点 i 是否被访问过
bool HasEdge[MAX_GRAPH_SIZE][MAX_GRAPH_SIZE];  // HasEdge[i][j]表示顶点 i
与 j 之间是否存在边
```

图的初始化

```
// 初始化图
// 图的顶点从 1 开始编号
void InitGraph()
{
    cout << "请输入图的顶点数："; cin >> VertexCount;
    int edge_count; cout << "请输入图的边数："; cin >> edge_count;
    cout << "请输入图中边的顶点对：" << endl;
    for (int i = 1; i <= edge_count; ++i)
    {
        int u, v; cin >> u >> v;
        HasEdge[u][v] = true;
        HasEdge[v][u] = true;
    }
}
```

第一个邻接顶点

```
// 找到编号为 v 的顶点的第一个邻接顶点
int FirstAdj(int v)
{
    if (v <= 0 || v > VertexCount) return 0;  // 检查参数合法性
    for (int u = 1; u <= VertexCount; ++u) {
        if (HasEdge[v][u]) return u;
    }
    return 0;
}
```

下一个邻接顶点

```
// 设 w 是 v 的邻接顶点，找到 v 的排在 w 后的下一个邻接顶点
int NextAdj(int v, int w)
{
    if (v <= 0 || v > VertexCount) return 0;  // 检查参数合法性
    if (w <= 0 || w > VertexCount) return 0;  // 同上
    for (int u = w + 1; u <= VertexCount; ++u) {
        if (HasEdge[v][u]) return u;
    }
    return 0;
}
```

深度优先遍历

```
// 对连通图从顶点 v 开始进行深度优先访问
void DepthFirstSearch(int v)
{
    memset(IsVisited, 0, sizeof(IsVisited));  // 清空访问标志
    stack<int> stk;
    stk.push(v);  // 从 v 顶点开始
    while (!stk.empty())
    {
        int vertex = stk.top(); stk.pop();  // 访问当前顶点
        if (IsVisited[vertex]) continue;
        cout << vertex; IsVisited[vertex] = true;
        for (int w = FirstAdj(vertex); w != 0; w = NextAdj(vertex, w)) {
            if (!IsVisited[w]) stk.push(w);
        }  // 依次将未访问的顶点加入栈
        if (!stk.empty()) cout << ' ';
    }
    cout << endl;
}
```

广度优先遍历

```
// 对连通图从顶点 v 开始进行广度优先访问
void BroadFirstSearch(int v)
{
    memset(IsVisited, 0, sizeof(IsVisited));  // 清空访问标志
    queue<int> que;
    que.push(v);  // 从 v 顶点开始
    while (!que.empty())
    {
        int vertex = que.front(); que.pop();  // 访问当前顶点
        if (IsVisited[vertex]) continue;
        cout << vertex; IsVisited[vertex] = true;
        for (int w = FirstAdj(vertex); w != 0; w = NextAdj(vertex, w)) {
            if (!IsVisited[w]) que.push(w);
        }  // 依次将未访问的顶点加入队列
        if (!que.empty()) cout << ' ';
    }
    cout << endl;
}
```

入口函数

```
int main()
{
    InitGraph();

    int random_vertex = rand() % VertexCount + 1;  // 随机顶点

    cout << "从顶点 1 开始的深度优先遍历序列：";
    DepthFirstSearch(1);
    cout << "从顶点" << random_vertex << "开始的深度优先遍历序列：";
    DepthFirstSearch(random_vertex);
    cout << "从顶点 1 开始的广度优先遍历序列：";
    BroadFirstSearch(1);
    cout << "从顶点" << random_vertex << "开始的广度优先遍历序列：";
    BroadFirstSearch(random_vertex);
}
```

五、实验结果及分析

程序的测试情况如下：

```
请输入图的顶点数：8
请输入图的边数：10
请输入图中边的顶点对：
1 2
1 3
2 4
2 5
3 6
3 7
4 8
5 8
6 8
7 8
从顶点 1 开始的深度优先遍历序列：1 3 7 8 6 5 2 4
从顶点 2 开始的深度优先遍历序列：2 5 8 7 3 6 1 4
从顶点 1 开始的广度优先遍历序列：1 2 3 4 5 6 7 8
从顶点 2 开始的广度优先遍历序列：2 1 4 5 3 8 6 7
```

程序的输出符合预期。
完成了本次实验的全部要求。