实验5: 图

确保把类定义写在`myGraph.h`中，类实现写在`myGraph.cpp`中，**不要修改文件名！！！**

`myGraph.h`文件中已经给出类定义和必要的函数，可以自行添加需要用到的函数，但不要变更已有的函数声明，测试文件只会调用已有的函数。

最终提交两个文件：`myGraph.h`和`myGraph.cpp`

注意：请务必使用在linux g++ std=c++11编译环境下可以使用的库函数

- 实现使用邻接表存储的无向图。边节点类`EdgeNode`、顶点节点类`VertexNode`和图类`MyGraph`，邻接表采用线性表存储，起点节点使用数组存储，同一起点节点的边节点使用链表存储，完成以下功能：
  - 边节点类的初始化和销毁、私有属性存取等基本功能
  - 顶点节点类的初始化和销毁、私有属性存取等基本功能
  - 图类的初始化和销毁、私有属性存取等基本功能
  - 图的带参数初始化`MyGraph(int, int, char*, int*, int*)`，参数列表为：
    - 顶点数目`int`
    - 边数目`int`
    - 顶点名称数组`char*`，长度等于顶点数目，不会出现重复的名称
    - 边的起点顶点下标数组`int*`（从0开始，无向图每条边只出现一次），长度等于边数目
    - 边的终点顶点下标数组`int*`（从0开始），长度等于边数目

> 建立的邻接表，起点顶点存储顺序应该与顶点名称数组中给出的顺序一致。
>
> 同一个起点的终点链表插入时应向链表**头部**插入。

  - 图的打印，`string printGraph()`，打印图的邻接表，返回一个`string`类型字符串，格式为：

```
“起点顶点名称：终点顶点1名称 终点顶点2名称....\n”

例如："A: B C D\nB: E F"
```

> 冒号后有空格，各个终点名称之间有空格，每个起点的最后一个终点名称后没有空格，而、是直接添加一个`\n`，最后一行末尾不加`\n`。终点名称的输出顺序按链表的存储顺序输出（即从`firstEdge`指向的开始输出）

- 实现图的深度优先搜索（DFS）和广度优先搜索（BFS）算法，输出对应的顶点序列：
  - 测试时，调用`string graph_DFS_Traverse()`和`string graph_BFS_Traverse()`函数，返回对应图的顶点序列`string`字符串，格式为：

```
“顶点名称+空格”（序列最后一个顶点后也要添加空格）

例如：“A B C D "
```

  - DFS与BFS结果不唯一，在遇到多种情况时，请按照邻接表中链表的存储顺序进行输出。

- 已经给出四个辅助用函数：`void BFS(int, int*, string&)`，`void DFS(int, int*, string&)`，`int getFirstNeighbor(int, int*)`，`int getNextNeighbor(int, int, int*)`。也可以不使用这四个辅助函数，根据自己的习惯自定义实现算法的中间函数。测试时不会调用这四个辅助函数，但请注意不要出现编译错误。

输入输出示例：

```
#include <iostream>
#include "myGraph.h"

using namespace std;

int main()
{
    int nodeNum = 9;
    int edgeNum = 10;
    char nodeList[9] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'};
    int edgeStartList[10] = {0, 0, 0, 1, 1, 4, 2, 3, 5, 7};
    int edgeEndList[10] = {1, 2, 3, 2, 4, 6, 5, 5, 7, 8};

    MyGraph g(nodeNum, edgeNum, nodeList, edgeStartList, edgeEndList);
    cout << g.printGraph() << endl;
    cout << g.graph_DFS_Traverse() << endl;
    cout << g.graph_BFS_Traverse() << endl;
    return 0;
}

/*
    控制台运行结果为：
    A: D C B
    B: E C A
    C: F B A
    D: F A
    E: G B
    F: H D C
    G: E
    H: I F
    I: H
    A D F H I C B E G
    A D C B F E H G I
*/
```