```
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   图的各种测试程序
//   Author：Melissa M.CAO
//   Belong：Section of software theory，School of Computer Engineering & Science，
Shanghai University
//   Version： 1.0
//////////////////////////////////////////////////////////////////////////////
//////////////////////////

#include "stdafx.h"

#include "MCAOTest.h"//不是模版类，无需内联方式编译
#include "CommonClass.cpp"
#include "AdjacencyMatrixGraph.cpp"
#include "AdjacencyListGraph.cpp"
#include "SeqList.cpp"
#include "MyHeap.cpp"
//#include "OrthogonalListGraph.cpp"

/*****************************************          第          七          章
************************************************************/
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
//    第七章--图章节的测试
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
void GraphTest()
{
    char selecttest = 'y';
    int term;
    while (selecttest == 'y' || selecttest == 'Y')
    {
        cout <<"请输入表示测试内容的数字：1--邻接矩阵表示的图的基本操作的测试；"
<< endl;
        cout << "                          2--邻接矩阵表示的图的遍历等测试；" <<
endl;
        cout << "                          3--邻接矩阵表示的图的最短路径各种算法
测试；" << endl;
        cout << "                          4--邻接矩阵表示的图的拓扑排序、关键路
径的测试；" << endl;
        cout <<"                         5--邻接矩阵表示的图的最小生成树的测试；
" << endl;
        cout                                                              <<
```

```
"------------------------------------------------------------------------"   <<
endl;
        cout <<"                              6--邻接表表示的图的基本操作的测试；"<<
endl;
        cout <<"                               7--邻接表表示的图的最短路径、生成树、
拓扑排序、关键路径等算法的测试；" << endl;
        cout                                                                  <<
"------------------------------------------------------------------------"   <<
endl;
        cout <<"                              8--无向图的多重邻接表表示方法的测试；"
<< endl;
        cout <<"                              9--有向图的十字链表表示方法的测试；"<<
endl;
        cout <<"                               10--；" << endl;
        cout                                                                  <<
"------------------------------------------------------------------------"   <<
endl;
        cout <<"                               11--第七章习题的测试；" << endl;
        cout <<"                               12--第七章实验题目的测试；" << endl;
        cout <<"                               13--第七章补充习题的测试；" << endl;
        cin >> term;
        switch (term)
        {
            case 1:
                AdjMatrixGraphTest();
            break;

            case 2:
                AdjMatrixGraphTravelTest();
            break;

            case 3:
                ShortestPathTest();
            break;

            case 4:
                TopologicalSortTest();
            break;

            case 5:
                MinSpanTreeTest();
            break;

            case 6:
```

```
            AdjListGraphCreate();
        break;

        case 7:
            AdjListGraphTest();
        break;

        case 8:
            //
        break;

        case 9:
            OrthogonalListGraphTest();
        break;

        case 10:

        break;

        case 11:
            ExcerciseOfChapSeven();
        break;

        case 12:
            ExperimentOfChapSeven();
        break;

        case 13:
            SumplementExcerciseOfChapSeven();
        break;

        default:
            cout <<"您输入的数字不在 1-13 这个范围内,找不到您指定的测试内容!
" << endl;
    }// end of switch
    cout << "您还想运行测试第七章--图章节的测试吗? (Y/N)";
    cin >> selecttest;
  } // end of while
}

////////////////////////////////////////////////////////////////////////////
//////////////////////
//建立图的邻接矩阵表示的测试
//测试数据:
```

```
/*
#
ABCDE#
@
ABACBCBDBECE@
1000
AB0.4CD1.8BC4.5DE4CE9@
*/
//////////////////////////////////////////////////////////////////////////////
//////////////////////
void AdjMatrixGraphTest()
{
    int c = 0;
    char cc, end, dd, a[20];
    float w, max;
    cout << "输入表示结束的结点值（如#)：";
    cin >> end;
    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);
    SeqList<char> v(20, c, a, 1);
    AdjacencyMatrixGraph<char, int> g1(c, 1, 1);
    g1.SetVertex(a, c);
    AdjacencyMatrixGraph<char, int> g3(c, 2, 1);
    g3.SetVertex(a, c);

    cout << "输入表示结束的边值（如$)：";
    cin >> end;
    cout << "请输入各条边(AB )：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        g1.InsertArc(cc, dd);
        g3.InsertArc(cc, dd);
        cin >> cc;
    }
    g1.Display();
    g3.Display();

    cout << "请输入表示无穷大的权值：";
    cin >> max;
    AdjacencyMatrixGraph<char, float> g2(c, 1, 2, max);
    g2.SetVertex(a, c);
    AdjacencyMatrixGraph<char, float> g4(c, 2, 2, max);
```

```
    g4.SetVertex(a, c);
    cout << "请输入各条边(AB2.1 )：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        cin >> w;
        g2.InsertArc(cc, dd, w);
        g4.InsertArc(cc, dd, w);
        cin >> cc;
    }

    g2.Display();
    g4.Display();

    cout << "请输入一个序号，我们帮你查找该结点的值：";
    cin >> c;
    cout << "第一个图中的第" << c << "个顶点为：" << g1.GetValue(c) << endl;
    cout <<"从" << g1.GetValue(1) <<"到" << g1.GetValue(c) <<"之间的边的权值为：
" << g1.GetWeight(g1.GetValue(1), g1.GetValue(c)) << endl;
    cout <<"从" << g2.GetValue(1) <<"到" << g2.GetValue(c) <<"之间的边的权值为：
" << g2.GetWeight(g2.GetValue(1), g2.GetValue(c)) << endl;
    cout << "请输入一个序号，我们帮你测试删除该结点后的邻接矩阵是否正确：";
    cin >> c;
    g1.DeleteVertex(g1.GetValue(c));
    g1.Display();
    g4.DeleteVertex(g4.GetValue(c));
    g4.Display();
}

//////////////////////////////////////////////////////////////////////////////
//////////////////////
//图的邻接矩阵表示的遍历等测试
//////////////////////////////////////////////////////////////////////////////
//////////////////////
void AdjMatrixGraphTravelTest()
{
    int c = 0, gt, wt;
    char cc, end, dd, a[20];
    float w, max = 0;
    cout << "输入表示结束的结点值（如#)：";
    cin >> end;

    CommonClass<char> inputobj;
```

```
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图）：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout <<"不处理混合图，请重新选择所建图的性质（1 表示无向图；2 表示有向图）：
";
        cin >> gt;
    }
    cout << "请输入所建图的边的性质（1 表示无权值图；2 表示带权图） ：";
    cin >> wt;
    while (wt != 1 && wt !=2)
    {
        cout << "不处理混合图，请重新选择所建图的边的性质（1 表示无权值图；2 表示
带权图）：";
        cin >> wt;
    }
    if (wt == 2)
    {
        cout << "请输入表示无穷大的权值：";
        cin >> max;
    }

    SeqList<char> v(20, c, a, 1);
    AdjacencyMatrixGraph<char, float> g1(c, gt, wt, max);
    g1.SetVertex(a, c);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;

    cout << "请输入各条边(AB 或 AB3.1)：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        if (wt == 1)
            g1.InsertArc(cc, dd);
        else
        {
            cin >> w;
            g1.InsertArc(cc, dd, w);
        }
```

```
        cin >> cc;
    }
    g1.Display();

    g1.DFTraverse();
    g1.BFTraverse();

    g1.IsConnected();
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////
//图的邻接矩阵表示的最短路径各种算法测试
/*
Test number1:
    @
    ABCDE@
    2
    10000
    #
    AB10
    BC50
    AD30
    AE100
    CE10
    DC20
    DE60
    #

    Test number2:
    @
    ABCDEFG@
    2
    10000
    #
    AB60
    AC50
    AD50
    BE-10
    CB-20
    CE10
    CG70
    DC-20
    DF-10
```

```
    EG30
    FG30
    #

    Test number3:
    @
    ABCD@
    2
    10000
    #
    AB54
    AC19
    AD12
    BC18
    CA15
    DA23
    DB6
    DC42
    #
    */
////////////////////////////////////////////////////////////////////////////////
////////////////////////
void ShortestPathTest()
{
    int c = 0, gt;
    char cc, end, dd, a[20];
    float w, max = 0;
    cout << "输入表示结束的结点值（如#)：";
    cin >> end;

    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图)：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout<<"不处理混合图，请重新选择所建图的性质(1 表示无向图;2 表示有向图)：
";
        cin >> gt;
    }

    cout << "请输入表示无穷大的权值：";
    cin >> max;
```

```
    SeqList<char> v(20, c, a, 1);
    AdjacencyMatrixGraph<char, float> g1(c, gt, 2, max);
    g1.SetVertex(a, c);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;

    cout << "请输入各条边(AB 或 AB3.1)：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        cin >> w;
        g1.InsertArc(cc, dd, w);
        cin >> cc;
    }
    g1.Display();

    cout <<"请输入最短路径的类型（1 表示单源点最短路径；2 表示多源点最短路径）：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout << "所选择的路径类型不存在！请重新输入（1 或 2）：";
        cin >> gt;
    }

    if (gt == 1)
    {
        cout << "请输入单源点顶点的值：";
        cin >> cc;
        g1.ShortestPath(2, c, cc);
    }
    else
        g1.Floyd();
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////
//拓扑排序、关键路径的测试；
////////////////////////////////////////////////////////////////////////////////
//////////////////////
void TopologicalSortTest()
{
```

```
    int c = 0;
    char cc, end, dd, a[20];
    float w, max = 0;
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;

    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入表示无穷大的权值：";
    cin >> max;

    SeqList<char> v(20, c, a, 1);
    AdjacencyMatrixGraph<char, float> g1(c, 2, 2, max);
    g1.SetVertex(a, c);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;

    cout << "请输入各条边(AB 或 AB3.1)：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        cin >> w;
        g1.InsertArc(cc, dd, w);
        cin >> cc;
    }
    g1.Display();
    g1.TopologicalSort();
//  g1.CriticalPathQuestion();
    g1.CriticalPath();
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////
//最小生成树的测试；
////////////////////////////////////////////////////////////////////////////////
//////////////////////
void MinSpanTreeTest()
{
    int c = 0;
    char cc, end, dd, a[20];
    float w, max = 0;
```

```
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;

    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入表示无穷大的权值：";
    cin >> max;

    SeqList<char> v(20, c, a, 1);
    AdjacencyMatrixGraph<char, float> g1(c, 1, 2, max);
    g1.SetVertex(a, c);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;

    cout << "请输入各条边(AB 或 AB3.1)：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        cin >> w;
        g1.InsertArc(cc, dd, w);
        cin >> cc;
    }
    g1.Display();
    g1.Kruskal();
    cout << "准备测试普里姆算法，请输入起始结点值（不是序号）：";
    cin >> cc;
    while (g1.FindVertex(cc) < 1)
    {
        cout << "您选择的结点不在图中，请重新输入起始结点值（不是序号）：";
        cin >> cc;
    }
    g1.Prime(cc);
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////
//建立图的邻接表表示的测试
/*
Test number1:
    @
    ABCDE@
```

```
2
2
#
AB10
BC50
AD30
AE100
CE10
DC20
DE60
#

Test number2:
@
ABCDEFG@
2
2
#
AB60
AC50
AD50
BE-10
CB-20
CE10
CG70
DC-20
DF-10
EG30
FG30
#

Test number3:
@
ABCD@
2
2
#
AB54
AC19
AD12
BC18
CA15
DA23
DB6
```

```
    DC42
    #
*/
////////////////////////////////////////////////////////////////////////////////
//////////////////////
void AdjListGraphCreate()
{
    int c = 0, gt, wt;
    char cc, end, dd, a[20];
    float w;
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;
    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图）：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout <<"不处理混合图，请重新选择所建图的性质（1 表示无向图；2 表示有向图）：
";
        cin >> gt;
    }
    cout << "请输入所建图的边的性质（1 表示无权值图；2 表示带权图）：";
    cin >> wt;
    while (wt != 1 && wt !=2)
    {
        cout << "不处理混合图，请重新选择所建图的边的性质（1 表示无权值图；2 表示
带权图）：";
        cin >> wt;
    }

    AdjacencyListGraph<char, int>  g1(a, c, gt, wt);
    AdjacencyListGraph<char, int>  g2(a, c, gt, wt);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;
    if (wt == 1) {
        cout << "请输入各条边(AB )：" << endl;
        cin >> cc;
        while (cc != end)
        {
            cin >> dd;
            g1.InsertArc(cc, dd, 1);
```

```
            g2.InsertArc(cc, dd, 2);
            cin >> cc;
        }
    }
    else
        if (wt == 2)
        {
            cout << "请输入各条边(AB2.1 )：" << endl;
            cin >> cc;
            while (cc != end)
            {
                cin >> dd;
                cin >> w;
                g1.InsertArc(cc, dd, w, 1);
                g2.InsertArc(cc, dd, w, 2);
                cin >> cc;
            }
        }
        else
        {
            cout << "图类型出错！";
            exit(1);
        }
    g1.Display();
    g2.Display();

    cout << "图中的第二个顶点为：" << g1.GetValue(1) << endl;
    cout << "从" << g1.GetValue(1) << "到" << g1.GetValue(2) << "之间的边的权值为：
" << g1.GetWeight(g1.GetValue(1), g1.GetValue(2)) << endl;
    g1.DeleteArc(1,2);
    g1.Display();
    g2.DeleteVertex(2);
    g2.Display();
    cout << "输入欲追加的顶点：";
    cin >> cc;
    g1.InsertVertex(cc);
    g1.Display();
    g1.DFTraverse();
    g1.BFTraverse();
    g2.Display();
    g2.DFTraverse();
    g2.BFTraverse();
}
```

```
//////////////////////////////////////////////////////////////////////////////
//////////////////////
//建立图的邻接表表示的测试
/*
Test number1:
    @
    ABCDE@
    2
    2
    #
    AB10
    BC50
    AD30
    AE100
    CE10
    DC20
    DE60
    #

    Test number2:
    @
    ABCDEFG@
    2
    2
    #
    AB60
    AC50
    AD50
    BE-10
    CB-20
    CE10
    CG70
    DC-20
    DF-10
    EG30
    FG30
    #

    Test number3:
    @
    ABCD@
    2
    2
    #
```

```
    AB54
    AC19
    AD12
    BC18
    CA15
    DA23
    DB6
    DC42
    #
*/
////////////////////////////////////////////////////////////////////////////////
//////////////////////
void AdjListGraphTest()
{
    int c = 0, gt, wt;
    char cc, end, dd, a[20];
    float w;
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;
    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图）：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout <<"不处理混合图，请重新选择所建图的性质（1 表示无向图；2 表示有向图）：
";
        cin >> gt;
    }
    cout << "请输入所建图的边的性质（1 表示无权值图；2 表示带权图）：";
    cin >> wt;
    while (wt != 1 && wt !=2)
    {
        cout << "不处理混合图，请重新选择所建图的边的性质（1 表示无权值图；2 表示
带权图）：";
        cin >> wt;
    }

    AdjacencyListGraph<char, int>  g1(a, c, gt, wt);
    AdjacencyListGraph<char, int>  g2(a, c, gt, wt);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;
```

```
if (wt == 1) {
    cout << "请输入各条边(AB )：" << endl;
    cin >> cc;
    while (cc != end)
    {
        cin >> dd;
        g1.InsertArc(cc, dd, 1);
        g2.InsertArc(cc, dd, 2);
        cin >> cc;
    }
}
else
    if (wt == 2)
    {
        cout << "请输入各条边(AB2.1 )：" << endl;
        cin >> cc;
        while (cc != end)
        {
            cin >> dd;
            cin >> w;
            g1.InsertArc(cc, dd, w, 1);
            g2.InsertArc(cc, dd, w, 2);
            cin >> cc;
        }
    }
    else
    {
        cout << "图类型出错！";
        exit(1);
    }
g1.Display();
g2.Display();

/*Test number:
@
ABCDEF@
1
2
#
AB6
AC1
AD5
BC5
BE3
```

```
CD5
CE6
CF4
DF2
EF6
#
*/
g2.Prim('A');
g2.Prim('E');
g2.Kruskal();

/*Test number1:
@
ABC@
1
2
#
AB26
BC-20
AC10
#

Test number2:
@
ABCDEF@
2
2
#
AB45
AC50
BC-15
AD20
DA10
DB10
DE35
BE20
EB30
BF15
FE-20
#
*/

cout <<"请输入最短路径的类型（1 表示单源点最短路径；2 表示多源点最短路径）：";
cin >> wt;
```

```
    while (wt != 1 && wt !=2)
    {
        cout << "所选择的路径类型不存在！请重新输入（1 或 2）：";
        cin >> wt;
    }
    if (wt == 1)
    {
        cout << "准备测试单源点最短路径算法，请输入起始顶点的序号和名称：";
        cin >> c;
        cin >> cc;
        g1.ShortestPath(1, c, cc);
        g2.ShortestPath(2, c, cc);
    }
    else        ;
        g1.Floyd();
}
//////////////////建立邻接表表示的图，测试第七章习题 14-15
void AdjListGraphPathBetweenTwoVertice()
{
    int c = 0, gt, wt, L;
    char cc, end, dd, a[20];
    float w;
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;
    cout << "请连续输入结点值，加结束标志（如 ABCD#）：";
    cin >> cc;
    while (cc != end)
    {
        a[c] = cc;
        c++;
        cin >> cc;
    }
    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图）：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout <<"不处理混合图，请重新选择所建图的性质（1 表示无向图；2 表示有向图）：
";
        cin >> gt;
    }
    cout << "请输入所建图的边的性质（1 表示无权值图；2 表示带权图）：";
    cin >> wt;
    while (wt != 1 && wt !=2)
    {
```

```
        cout << "不处理混合图，请重新选择所建图的边的性质（1 表示无权值图；2 表示带权图）：";
        cin >> wt;
    }

    AdjacencyListGraph<char, int>  g1(a, c, gt, wt);
    AdjacencyListGraph<char, int>  g2(a, c, gt, wt);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;
    if (wt == 1) {
        cout << "请输入各条边(AB )：" << endl;
        cin >> cc;
        while (cc != end)
        {
            cin >> dd;
            g1.InsertArc(cc, dd, 1);
            g2.InsertArc(cc, dd, 2);
            cin >> cc;
        }
    }
    else
        if (wt == 2)
        {
            cout << "请输入各条边(AB2.1 )：" << endl;
            cin >> cc;
            while (cc != end)
            {
                cin >> dd;
                cin >> w;
                g1.InsertArc(cc, dd, w, 1);
                g2.InsertArc(cc, dd, w, 2);
                cin >> cc;
            }
        }
        else
        {
            cout << "图类型出错！";
            exit(1);
        }

        end = 'Y';
        while (end == 'Y' || end == 'y')
        {
```

```
            cout << "请输入两个结点，以判断它们之间是否存在路径！";
            cin >> cc >> dd;
            cout <<"如果要限制为简单路径且制定路径长度，请输入该长度，否则，请输入-100：";
            cin >> L;
            if (L == -100)
            {
                cout << "请输入欲采用的方法，D 或 d 代表深度优先搜索[两种不同的解法]，B 或 b 代表广度优先搜索[两种不同的解法]！";
                cin >> end;
            }
            else
                end = 'D';
            c = g1.ExistPath(cc, dd, end, L);
            if (c == 0)
                cout << "两顶点之间没有路径！" << endl;
            else
                if (c == 1)
                    cout << "两顶点之间存在路径！" << endl;
                else
                    cout << "出错啦！" << endl;
            cout << "继续测试请输入 Y 或 y，否则退出该测试！";
            cin >> end;
        }
 }

void OrthogonalListGraphTest()
{
//  OrthogonalListGraph<char, double> g(1);
}

/************************************** 第 七 章 习 题 测 试 开 始 ***********************************************/
void ExcerciseOfChapSeven()
{
    int num;
    char selecttest = 'y';

    //先构造数据表

    while (selecttest == 'y' || selecttest == 'Y')
    {
        cout << "请输入题号（1-32）：" << endl;
        cin >> num;
```

```
        switch (num)
        {
            case 1:
            case 2:
            case 3:
            case 4:
            case 6:
            case 32:
                cout <<"书面作业，无需验证！若有兴趣，请输入数据，调用相应的算法！
" << endl;
            break;

            case 5:
                cout << "基本操作中实现，即选择操作 5，给出测试数据即可看到答案！
" << endl;
            break;

            case 7:
            case 8:
            case 9:
                cout << "基本操作中实现，即选择操作 3，给出测试数据即可看到答案！
" << endl;
            break;

            case 10:
            case 11:
                cout << "基本操作中实现，即选择操作 4，给出测试数据即可看到答案！
" << endl;
            break;

            case 12://无向图的多重邻接表基本操作
                //
            break;

            case 13://有向图的十字链表基本操作
                //
            break;

            case 14:
            case 15:
            case 19://判断两顶点间是否存在长度为 A 的简单路径--邻接表
                //测试数据：
                cout << "邻接表表示的图，习题 7.14、15、19，判断两个结点之间有无
```

```
【长度为A的】路径的测试；" << endl;
                AdjListGraphPathBetweenTwoVertice();
            break;

            case 16: //利用栈，进行图的深度优先非递归遍历
                //测试数据：
            break;

            case 17: //判断有向图中是否有回路
                //
            break;

            case 20: //邻接矩阵的有向图，求i和j之间不含回路的长度为k的路径数
                //
            break;

            case 21: //求有向图中所有简单回路
                //
            break;

            case 22: //求强连通分量
                //
            break;

            case 23: //修改Prim算法，求图的最小生成森林
                //
            break;

            case 25: //求有向无环图的根
                Excercise25();
            break;

            case 26: //求有向无环图中每个顶点出发的最长路径
            break;

            case 27: //求有向无环图的最长路径
            break;

            case 18: //对有向无环图重排顶点序号--邻接矩阵变为下三角矩阵
            case 24: //求有向无环图顶点赋予序号以满足条件
            case 28: //利用深度优先搜索求关键路径
            case 31: //利用深度优先搜索求拓扑排序--邻接表
                Excercise2831(num);
            break;
```

```
            case 29:
                cout << "基本操作中实现，即选择操作 7，给出测试数据即可看到答案！
" << endl;
            break;

            case 30: //十字链表中的 Bellman-Fort 算法
                //测试数据 4--一般：
            break;

            case 33: //判断是否有桥
                //
            break;

            default:
                cout <<"您输入的数字不在 1-33 这个范围内,找不到您指定的测试内容！
" << endl;

        } // end of switch

        cout << "您还想运行第七章(图)的习题测试吗？（Y/N）";
        cin >> selecttest;

    } // end of while (selecttest == 'y' || selecttest == 'Y')
}

////////////////////////////////////////////////////////////////////////////////
//////////////
/*
```

7.25　若有向无环图中存在一个顶点 r，如果在 r 和图中其他所有顶点之间均存在由 r 出发的有向路径，则称该 DAG 有根。

试编写求有向无环图中根的算法。有根，返回根的序号，否则，返回-1。图类型不正确，返回-2

注意：　该算法要求不能有环，否则会发生误判。

测试数据 1：

```
#
ABCDE#
2
1
@
ABACBDCE@
```

测试数据 2：
#
ABCD#
2
1
@
ABADBCDBCD@

测试数据 3：
#
ABCDEF#
2
1
@
ABACBDFCFD@

测试数据 4：
#
ABCDEF#
2
1
@
ABACBDCEFA@
*/
```
////////////////////////////////////////////////////////////////////////////////
/////////////////
void Excercise25()
{
    //先建立图
    int c = 0, gt, wt, L;
    char cc, end, dd, a[20];
    float w;
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;
    cout << "请连续输入结点值，加结束标志（如 ABCD#）：";
    cin >> cc;
    while (cc != end)
    {
        a[c] = cc;
        c++;
        cin >> cc;
    }
    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图）：";
    cin >> gt;
```

```
    while (gt != 1 && gt !=2)
    {
        cout<<"不处理混合图，请重新选择所建图的性质(1 表示无向图;2 表示有向图)：
";
        cin >> gt;
    }
    cout << "请输入所建图的边的性质（1 表示无权值图；2 表示带权图)：";
    cin >> wt;
    while (wt != 1 && wt !=2)
    {
        cout << "不处理混合图，请重新选择所建图的边的性质（1 表示无权值图；2 表示
带权图)：";
        cin >> wt;
    }

    AdjacencyListGraph<char, int>  g1(a, c, gt, wt);

    cout << "输入表示结束的边值（如$)：";
    cin >> end;
    if (wt == 1) {
        cout << "请输入各条边(AB )：" << endl;
        cin >> cc;
        while (cc != end)
        {
            cin >> dd;
            g1.InsertArc(cc, dd, 1);
            cin >> cc;
        }
    }
    else
        if (wt == 2)
        {
            cout << "请输入各条边(AB2.1 )：" << endl;
            cin >> cc;
            while (cc != end)
            {
                cin >> dd;
                cin >> w;
                g1.InsertArc(cc, dd, w, 1);
                cin >> cc;
            }
        }
        else
        {
```

```
            cout << "图类型出错！";
            exit(1);
        }

    int result = g1.GetRoot();
    if (result == -1)
        cout << "该有向无环图无根！" << endl;
    else
        if (result == -2)
            cout << "该图不是有向无环图！" << endl;
        else
            cout << "该有向无环图有根！为" << g1.GetValue(result) << endl;
}

//////////////////////////////////////////////////////////////////////////////
//////////////
/*
7.18   给每个顶点编号，使其满足条件：邻接矩阵变为下三角矩阵。----就是满足逆拓扑排序的编号方法。
7.24   给每个顶点编号，使其满足条件：若 i 到 j 有一条弧，则 i 的所编号一定小于 j 的所编号。--就是满足拓扑排序的编号方法。
7.28   利用深度优先遍历有向图实现求关键路径算法。
7.31   邻接表存储的有向图，按深度优先搜索策略拓扑排序
注意：  该算法要求不能有环，否则会发生误判。

Test number1:
    @
    ABCDE@
    2
    2
    #
    AB10
    BC50
    AD30
    AE100
    CE10
    DC20
    DE60
    #

    Test number2:
    @
    ABCDEFG@
    2
```

2
#
AB60
AC50
AD50
BE-10
CB-20
CE10
CG70
DC-20
DF-10
EG30
FG30
#

Test number3:
@
ABCD@
2
2
#
AB54
AC19
AD12
BC18
CA15
DA23
DB6
DC42
#

Test number4:
 @
abcdefg@
2
2
#
ab8
ad4
ae5
bc3
cg6
de1
eb2

```
    ec7
    ef2
    fc3
    fg9
    #
*/
////////////////////////////////////////////////////////////////////////////////
/////////////////
void Excercise2831(int excersiseno)
{
    int c = 0, gt, wt;
    char cc, end, dd, a[20];
    float w;
    cout << "输入表示结束的结点值（如#）：";
    cin >> end;
    CommonClass<char> inputobj;
    c = inputobj.InputDataInArray(a, 2, end);

    cout << "请输入所建图的性质（1 表示无向图；2 表示有向图）：";
    cin >> gt;
    while (gt != 1 && gt !=2)
    {
        cout <<"不处理混合图，请重新选择所建图的性质（1 表示无向图；2 表示有向图）：
";
        cin >> gt;
    }
    cout << "请输入所建图的边的性质（1 表示无权值图；2 表示带权图）：";
    cin >> wt;
    while (wt != 1 && wt !=2)
    {
        cout << "不处理混合图，请重新选择所建图的边的性质（1 表示无权值图；2 表示
带权图）：";
        cin >> wt;
    }

    AdjacencyListGraph<char, int>  g1(a, c, gt, wt);
    AdjacencyListGraph<char, int>  g2(a, c, gt, wt);

    cout << "输入表示结束的边值（如$）：";
    cin >> end;
    if (wt == 1) {
        cout << "请输入各条边(AB )：" << endl;
        cin >> cc;
        while (cc != end)
```

```
    {
        cin >> dd;
        g1.InsertArc(cc, dd, 1);
        g2.InsertArc(cc, dd, 2);
        cin >> cc;
    }
}
else
    if (wt == 2)
    {
        cout << "请输入各条边(AB2.1 )：" << endl;
        cin >> cc;
        while (cc != end)
        {
            cin >> dd;
            cin >> w;
            g1.InsertArc(cc, dd, w, 1);
            g2.InsertArc(cc, dd, w, 2);
            cin >> cc;
        }
    }
    else
    {
        cout << "图类型出错！";
        exit(1);
    }
g1.Display();
g2.Display();

c = excersiseno;
if (excersiseno == 28)
{
    g1.DFSCriticalPath();
    g2.DFSCriticalPath();
}
else
    if ((excersiseno == 31) || (excersiseno == 24) || (excersiseno == 18))
    {
        g1.DFSTopologicalSort(&c);
        c = excersiseno;
        g2.DFSTopologicalSort(&c);
    }
    else
    cout << "习题序号错误，赶紧检查一下是否有鬼！" << endl;
```

```
}

/****************************************** 第 七 章 习 题 测 试 结 束 ******************************************/

////////////////////////////////////////////////////////////////////////////////
//////////////////////
//  第七章实验题目的测试
////////////////////////////////////////////////////////////////////////////////
//////////////////////
void ExperimentOfChapSeven()
{
    int num;
    char selecttest = 'y';

    //先构造数据表

    while (selecttest == 'y' || selecttest == 'Y')
    {
        cout << "请输入题号 (1-3)：" << endl;
        cin >> num;

        switch (num)
        {
            case 1:
                cout << "基本操作中实现，即选择操作 1（邻接矩阵）或 6（邻接表）即可！" << endl;
            break;

            case 2:

            break;

            case 3:

            break;

            default:
                cout << "您输入的数字不在 1-3 这个范围内，找不到您指定的测试内容！" << endl;

        } // end of switch
```

```
        cout << "您还想运行第七章(图)的实验测试吗？（Y/N）";
        cin >> selecttest;

    } // end of while (selecttest == 'y' || selecttest == 'Y')
}

////////////////////////////////////////////////////////////////////////////////
////////////////////////
//  第七章补充习题测试
////////////////////////////////////////////////////////////////////////////////
////////////////////////
void SumplementExcerciseOfChapSeven()
{
    int num;
    char selecttest = 'y';

    //先构造数据表

    while (selecttest == 'y' || selecttest == 'Y')
    {
        cout << "请输入题号（1-3）：" << endl;
        cin >> num;

        switch (num)
        {
            case 1:

            break;

            case 2:

            break;

            case 3:

            break;

            default:
                cout << "您输入的数字不在 1-3 这个范围内，找不到您指定的测试内容！
" << endl;

        } // end of switch
```

```
        cout << "您还想运行第七章(图)的补充习题测试吗？（Y/N）";
        cin >> selecttest;

    } // end of while (selecttest == 'y' || selecttest == 'Y')
}
```