****

**Tongji University**

**离散数学实验报告**

**003-**求关系的自反、对称和传递闭包

专 业： 软 件 工 程

指导教师： 唐 剑 锋

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **求关系的自反、对称和传递闭包**
   1. **实验概述**
      1. **实验要求**

本次实验要求以程序设计的方法实现求关系的自反，对称和传递闭包。关系由关系矩阵表示，用户手动输入关系矩阵的行列数后输入关系矩阵，选择求关系矩阵的自反/对称/传递闭包操作，程序化实现操作后将关系矩阵打印出来。

* + 1. **实验原理**

根据关系自反，对称和传递闭包的定义可以得到求解方法的原理如下。

设R为A上的关系,则有：

1. r(R)=R∪R0
2. s(R)=R∪R−1
3. t(R)=R∪R2∪R3∪…

对以矩阵表示的关系，其自反闭包只要将矩阵的主对角线全部置为1，对称闭包则由关系矩阵加上其转置矩阵得到（逻辑加）。

1. **整体设计思路**
   1. **对实验要求的分析**

本次实验要求重点在于用C++语言实现求关系的自反，对称和传递闭包。具体可以将实验细化为几个模块，包括：

1. 在基本main函数中实现用户端数据的输入与错误处理；
2. 通过分函数实现求关系自反，对称和传递闭包操作；
3. 考虑到对称闭包求解需要求矩阵转置，故另设函数求解转置矩阵；
4. 在output函数中实现矩阵的逐行输出；
   1. **程序流程图**

结束

输入关系矩阵行列

合法？

否

是

输入需要的操作

打印迷宫

求解自反闭包

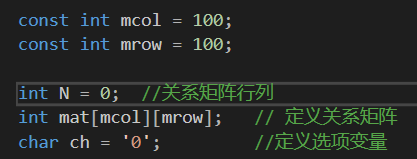
求解对称闭包

求解传递闭包

开始

1. **数据结构的使用**
   1. **顺序表数据结构的使用**

本次实验使用顺序表中的静态数组数据结构，利用全局二维数组存储关系矩阵，避免了函数形参过分复杂，同时也方便对矩阵内部进行增删查改而不需要进行多余的遍历操作，因此可以降低时间复杂度。



* 1. **常变量的使用**

本次实验由于变量较多且形式复杂，因此为了解决调用不当或者错误使用的风险等，笔者定义了两个常变量mcol和mrow，作为关系矩阵的初始行列设定。

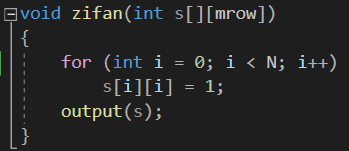
因为常变量具有程序中改变非法，存储稳定等特性，可以在变量较多的程序中降低错误风险。



1. **核心算法分析与优化**
   1. **求关系矩阵的自反闭包**

本次实验求关系矩阵的自反闭包即将关系矩阵主对角线元素全部置为1，通过zifan()函数进行实现，并在函数末尾调用output()函数进行关系矩阵的打印。

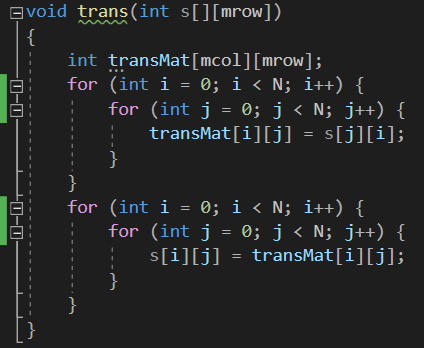
具体函数实现如下：



* 1. **求关系的对称闭包**

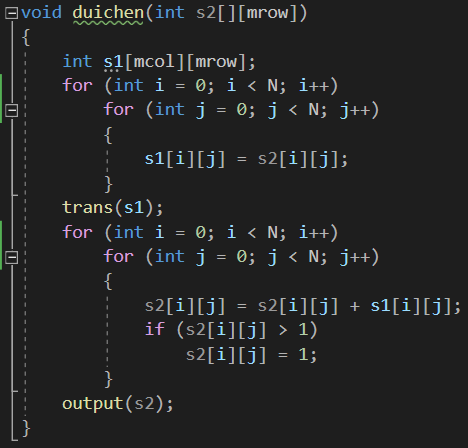
对于程序化实现求关系的对称闭包的问题，可以通过关系矩阵逻辑加其转置矩阵进行得到。

而出于方便考虑，则设定转置函数来求矩阵的转置，即设定新二维数组变量transMat，遍历原关系矩阵，并将每个位置transMat[i][j]置为s[j][i]，此时transMat中即存储着原关系矩阵的转置矩阵，再次遍历赋值即可。



而在duichen()函数中新定义二维数组s1，使其转置成为原关系矩阵的转置矩阵。遍历矩阵，并将两矩阵相同位置的数相加，若大于1，则赋值为1（关系矩阵中不存在大于1的数），由此得到关系矩阵的对称闭包。

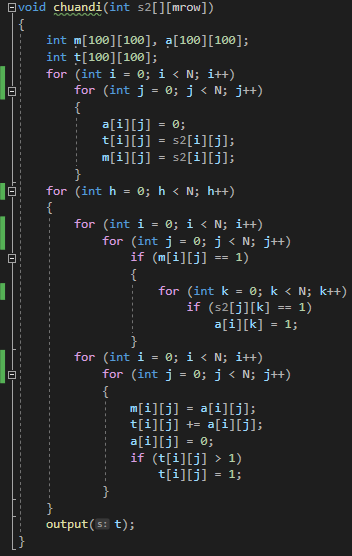
在函数末尾调用output（）函数，打印结果矩阵。



* 1. **求关系的传递闭包**

本次实验中求关系的传递闭包利用了传递闭包公式t(R)=R∪R2∪R3∪…；而对于公式的实现则依托于for循环的嵌套。

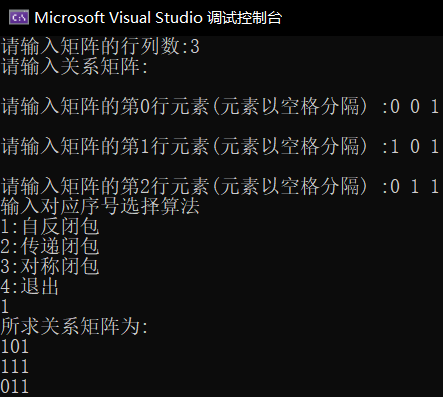
1. 首先在函数中定义3个新的二维静态数组，通过遍历将数组a全置为0，将t和m的值置为的值；
2. 因为传递闭包需要关系R一直合并置Rn有n次求并集操作，故外层大循环嵌套N次（N为关系矩阵行列数）
3. 在大循环中先遍历原关系矩阵，若m矩阵为1时，利用循环N次求得Rk（k=2,3,…）
4. 此后将关系矩阵求并集置入二维数组t，当t的值大于1时，将t置为1.
5. 在函数末尾调用output函数，打印传递闭包矩阵。

****

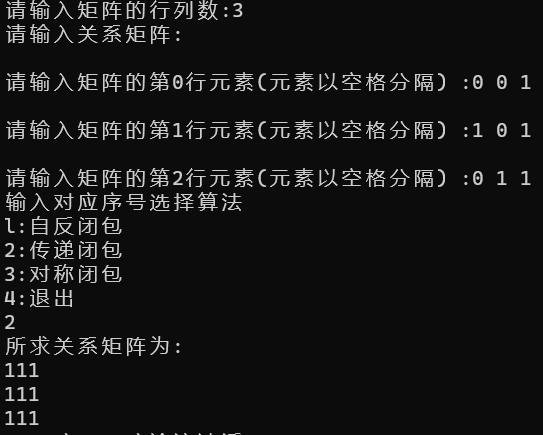
1. **心得体会**

通过本次实验，笔者收获颇丰，主要体现在以下几个方面：

1. 通过实验加深了对于关系矩阵自反、对称和传递闭包的理解，同时也应用了求三种闭包的方法，进一步切身体会了闭包的由来和求解方法；
2. 通过编写代码进一步熟练了C++语言下对于各种离散数学关系的实现，通过定义静态数组、常变量等数据结构，学会了利用不同结构解决问题的方法。
3. 实验中遇到了函数形参是否会改变，输入缓冲区清空语句的调用位置等问题，都通过分步调试的方式逐个解决。本次实验也强化了笔者调试程序改正错误的能力，熟练了分步调试，逐过程调试的调试方法。
4. **程序测试**
   1. **求关系的自反矩阵**

****

* 1. **求关系的传递矩阵**



* 1. **求关系的对称矩阵**

