《离散数学》课程实验报告6 用Warshall算法求关系的传递闭包

**1.项目要求**

用户从键盘处输入一个关系，要求程序输出该关系用集合表示的传递闭包。

**2.实验原理**

Warshall算法流程大致如下：

1、置新矩阵M=AR（AR表示集合A的二元关系R的集合）；

2、置j=1；

3、对所有i(i=1,2,…,n, n是集合A中元素的个数)，若M[i][j]=1,则对k=1,2,…,n,置M[i][k]=M[i][k]+M[j][k];

4、j=j+1;

5、若j<=n,则转到步骤3进行循环，否则退出循环。

**3.实现思路及重要函数**

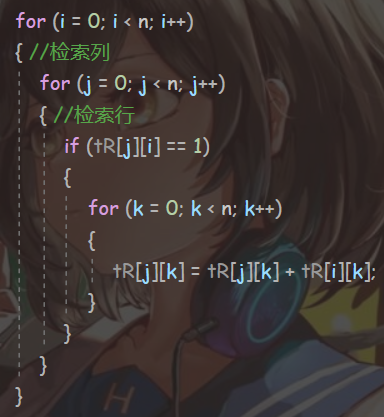
**3.1.输入处理流程**

在程序的开始要求用户初始化全集和二元关系，这里需要调用init\_aggregation函数和init\_BinaryRelation()函数。这两个函数处理流程相似，都是先从用户处获取待输入元素（关系）的数量并进行输入错误处理，若输入一个负整数或字符则要求用户重新输入。然后动态内存申请相应大小的空间，再从用户处获取具体元素（关系）的输入，每一个关系存储在结构体BR中。

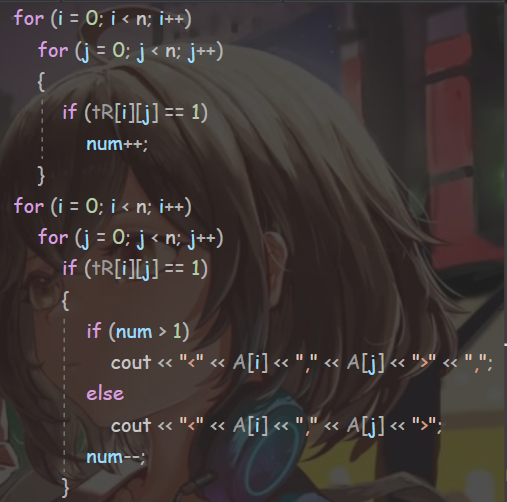
**3.2.开辟并初始化传递闭包矩阵**

这一步比较简单，使用动态内存申请的方式开辟一个二维bool型大小为n\*n的（为了防止溢出，我将大小设置为(n+1)\*(n+1)）数组便于逻辑加的运算，并将数组元素全部初始化为0.

**3.3.调用Warshall算法函数**

Warshall算法再程序中由函数void Warshall(char\*& A, BR\*& R, bool\*\*&tR)实现，其中各个参数表示意义如下：A代表集合数组，R代表关系数组，tR代表传递闭包数组。函数的第一步是将以数组形式存储的关系先写到传递闭包矩阵中，这里需要调用fun函数，这个函数的作用是将用字符表示的元素转化为原集合中的索引数字。最后就是对已经写入了初始关系的矩阵tR用Warshall算法计算传递闭包。检索每一列的每一行的元素，倘若该元素为1，则将与目前检索的与列数相同的行的所有元素加到这一行（如实验原理所示），关键代码如右图（该代码段中i、j的顺序与实验原理中恰好相反）。

**3.4. 将传递闭包以集合形式输出**

通过调用Warshall函数，传递闭包已经以矩阵的形式存储在tR中，现在需要一个函数把以矩阵形式表示的传递闭包以集合形式输出。这里需要调用translation\_output(char\*& A, bool\*\*& tR)函数，其中A是原集合，tR是传递闭包矩阵。先遍历一次传递闭包矩阵，记录其中为1元素的个数num，便于控制最后输出传递闭包集合时控制集合中元素分隔符的输出，避免输出最后一个元素后仍输出分隔符“，”。然后再一次遍历tR数组，当遍历到1时，输出该元素位置对应的关系，若num>1,则输出“，”作为集合间元素的分隔符；否则说明已经遍历到最后一个元素，不输出“，”，输出完成后num--。关键代码段如右图。

**4.心得体会**

通过这次作业，我对Warshall算法的原理和代码实现有了比较深刻的了解。和大作业三中传统求传递闭包的方式对比，时间复杂度由O(n^4)降低到了O(n^3)，大大降低了时间复杂度。同时我也积累了根据文字或伪代码描述写出一个较复杂算法的能力。自行解决了translation\_output函数中输出最后一个元素后会再输出“，”的问题，同时提高了程序的健壮性，当用户应该输入正整数时若输入字符或负数，程序会提示用户重新进行正确的输入。整个程序效果如下。

