关系闭包的计算

小组成员：甘明浩、崔宇新、丘善烨、唐永杰

**一、实验目的**

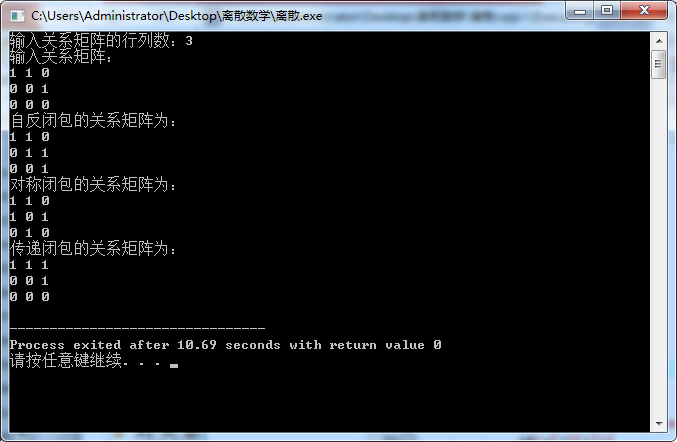
掌握从关系矩阵中求关系的自反闭包、对称闭包和传递闭包的方法。

**二、实验内容**

输入：一个关系的关系矩阵（键盘输入或从文件读入）

输出：此关系的自反闭包、对称闭包和传递闭包所对应的关系矩阵

**三、实验结果**



同预期一致，可以求出n阶关系矩阵的自反闭包、对称闭包、传递闭包。

**四、代码分析（详见源文件）**

**1. 读取矩阵**

int n;

cout<<"输入关系矩阵的行列数：";

cin>>n;

int \*\*a=new int \* [n];

for(int i=0;i<n;i++)

{

a[i]=new int [n];

}

cout<<"输入关系矩阵："<<endl;

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

cin>>a[i][j];

}

分析：接受到矩阵的行列值n后，简单地通过一个二级指针以及new与delete运算符动态分配一片存储空间用于存储二维数组（关系矩阵），再从键盘中读取关系矩阵的值存入二维数组中。

**2. 自反闭包**

int \*\*b=new int \* [n];

for(int i=0;i<n;i++)

{

b[i]=new int [n];

}

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

b[i][j]=a[i][j];

b[i][i]=1;//关键

}

cout<<"自反闭包的关系矩阵为："<<endl;

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

cout<<b[i][j]<<' ';

cout<<endl;

}

delete []b;

分析：同1，再动态分配生成一个二维数组b，用于存储自反闭包的关系矩阵。自反闭包的关系矩阵可以通过原关系矩阵M R与单位矩阵I（主对角线全为1）的对应元素相加得到，即主对角线的元素值全为1，实现较为简单。

**3. 对称闭包**

int \*\*c=new int \*[n];

for(int i=0;i<n;i++)

{

c[i]=new int [n];

}

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

c[i][j]=a[i][j];

}

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

if(c[i][j]==1) c[j][i]=1;

}

}

cout<<"对称闭包的关系矩阵为："<<endl;

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

cout<<c[i][j]<<' ';

cout<<endl;

}

delete []c;

分析：同1，再动态分配生成一个二维数组c，用于存储对称闭包的关系矩阵。对称闭包的关系矩阵可以通过原关系矩阵M R与原关系的逆运算**R-1**的关系矩阵M R**-1** （转置矩阵MR**T**）的对应元素相加得到，即元素关于主对角线对称。

**4. 传递闭包**

int \*\*d=new int \*[n];

for(int i=0;i<n;i++)

{

d[i]=new int [n];

}

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

d[i][j]=a[i][j];

}

for(int i = 0; i < n; i++)

for(int j = 0; j <n; j++)

if(a[j][i]){

for(int k=0;k<n;k++)

d[j][k]+=a[i][k];

}

cout<<"传递闭包的关系矩阵为："<<endl;

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++){

if(d[i][j]>1) d[i][j]=1;

cout<<d[i][j]<<' ';}

cout<<endl;

}

分析：同1，再动态分配生成一个二维数组d，用于存储传递闭包的关系矩阵。传递闭包的关系矩阵

1. 可以通过R的关系矩阵、**R2** 的关系矩阵、……**R n** 的关系矩阵的对应元素相加得到。

即**Mt= MR+ MR2+ MR3+…+ MRn**

可以通过定义一个函数进行n次关系的幂运算（矩阵的乘法）相加后得到，但这样代码实现较为复杂且繁琐。

1. Floyd-Warshall算法。即代码标红部分。这应该略微属于课外内容，但代码实现十分方便简洁。具体操作如下：

按列号顺序对M的每一列元素从上到下依次扫描。如果当前扫描的是第 i 列，那么当遇到1时，将1所对应的行加上第i 行,结果记录到刚才1所对应的行。

for i ← 1 to n do

for j ← 1 to n do

if a[j,i]=1 then

for k ← 1 to n do

d[j,k]=d [j,k]+ a [i,k]

**五、实验感受**

通过这一次的实验，既加强了我们小组成员对离散数学中关系的自反性、对称性、传递性等性质以及闭包运算的理解与掌握，又加强了我们的动手与学习能力，找到了一种在编程中构造传递闭包的简便的方法——Floyd-Warshall算法，同时学会了很多课堂上学不到的知识，也了解到了离散数学的相关概念与编程实现的关系。可惜这次的实验不算特别复杂，没能让我们更深刻地体会到离散数学与编程的关系，希望还能有机会做一些小组合作的编程实验。