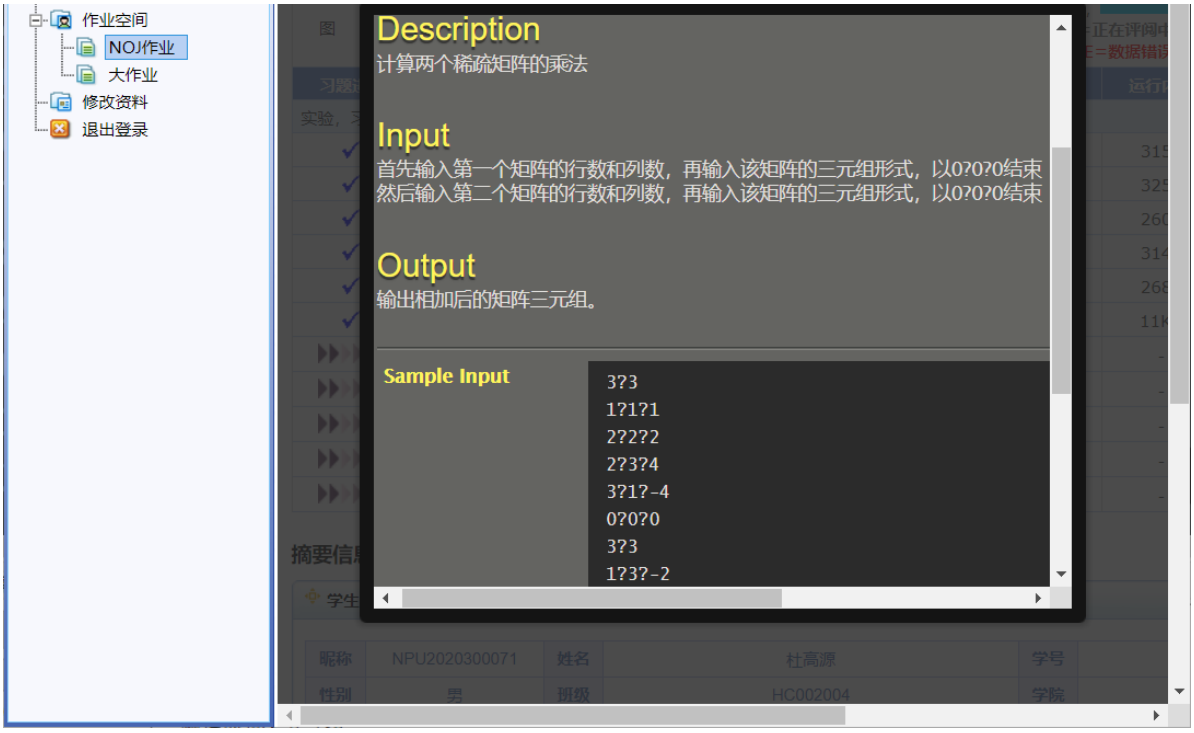
noj实验6报告

0.0 题目//需求分析



需求分析：即输入两个矩阵，计算它们 的乘积（乘积是线性代数知识）

1.0 实验思路

第一步 完成初始化以及输入操作，注意设计0 0 0 作为输入终止条件

第二步 乘法代码，按照老师说的算法，做前行乘以后行的操作，所乘得的结果是分量，存储在一个临时辅助数组里，然后继续后移，重复执行操作，然后将分量累加起来。

第三步 输出

听说printf还能“重载”？皮了一下（printf）。

2.0 代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXN 200//三元组最大个数
typedef struct
{
    int row,col;
    int elem;
}Triple;
typedef struct
{
    Triple data[MAXN];
    int rpos[MAXN];//位置辅助数组
```

```
    int m,n,len;
}TripleMatrix;
void intimatrix(TripleMatrix *M)
{
   int num[MAXN];
   scanf("%d%d",&(M->m),&(M->n));
   int x1,x2,x3;
   int i,j;
   M->len=0;
   while(!0)
   {
       scanf("%d%d%d",&x1,&x2,&x3);//输入三元组
       if(x1==0&&x2==0&&x3==0)
       break;//输入一个矩阵三元组时的终止条件
       M->data[M->len].row=x1;
       M->data[M->len].col=x2;
       M->data[M->len].elem=x3;//分别赋值
       M->len++;
   }
     if(M->len)
   {
     for(i=0;i<=M->m;i++)        //列数是从1开始的
     num[i]=0;
     for(j=0;j<M->len;j++)       //data从0开始记的
     {
      num[M->data[j].row]++;     //与转置不同，这里找的是每一种行值有几个数。
     }                           //rpos是每一种行值的第一个非0元素在三元组中的位置
     M->rpos[0]=0;
     for(i=1;i<=M->m;i++)
     {
       M->rpos[i]=M->rpos[i-1]+num[i-1];
     }
      M->rpos[i]=M->len;     //这里的i是比总行数大1的，标志着A的最后一行的位置
   }
     return;
}
void multimatrix(TripleMatrix *A,TripleMatrix *B,TripleMatrix *C)
{
    int arow,brow,ccol;
    int i,j;
    int ctemp[MAXN];
    for(arow=1;arow<=A->m;arow++)
    {
        C->rpos[arow]=C->len;
        for(i=A->rpos[arow];i<A->rpos[arow+1];i++)    //i是该行列在A中对应的位置
        {
            memset(ctemp,0,sizeof(ctemp));
            brow=A->data[i].col;
            for(j=B->rpos[brow];j<B->rpos[brow+1];j++)  //j是该行列在B中对应的位置
          {
              ccol=B->data[j].col;
              ctemp[ccol]+=A->data[i].elem*B->data[j].elem;
          }
          for(ccol=0;ccol<=C->n;ccol++)
          {
                    if (ctemp[ccol]!=0)
                  {
                     C->data[C->len].row=arow;
```

```
                    C->data[C->len].col=ccol;
                    C->data[C->len].elem=ctemp[ccol];
                    C->len++;
                }
            }
        }
    }
    return;
}
void print(TripleMatrix *C)//输出函数
{
  int i;
  for(i=0;i<C->len;i++)
  printf("%d %d %d\n",C->data[i].row,C->data[i].col,C->data[i].elem);
  return;
}
int main()
{
   TripleMatrix A,B,C;
   intimatrix(&A);
   intimatrix(&B);  //√
   C.m=A.m;
   C.n=B.n;
   C.len=0;//初始化C
   if (A.m==B.n

)//相乘条件
   multimatrix(&A,&B,&C);
   print(&C);//√
   return 0;
}
```

4.0 心得

实现这个乘法操作真的是麻烦而不太难的事情，这次又是在反复在调指针的bug。

在纸上的分析确实很重要，刚开始毫无思路，到后来纸上分析后，就简化为双向链表的存取及运算。

注：参考了网上资料思路。