暨南大学本科实验报告专用纸

课程名称 高级语言程序设计实验 成绩评定

实验项目名称 矩阵运算的函数封装 指导教师 张鑫源

实验项目编号 ⑨ 实验项目类型 实验地点 家中

学生姓名 位雨昕 学号 2019051112

学院 智能科学与工程学院 系 专业 信息安全

实验时间 2020 年 4 月 29 日 上午～4 月 30 日 下午 温度 ℃湿度

（一）实验目的

1．进一步了解 Visual Studio 的使用以及 C 语言程序的结构；

2．接触 C 语言中的常用函数、掌握其使用方法；

3．熟练掌握循环语句、二维数组的使用；

4．锻炼个人的编程操作能力。

（二）实验内容和要求

内容：

对于矩阵 A, B 常规的矩阵操作有 A^T，A+B，A-B，A×B，A^n，分别用函数封装上述矩阵操作。

要求：

1. A 和 B 为程序自定义；
2. 设置合理的输出展示封装效果。

（三）主要仪器设备

仪器：计算机

实验环境：Visual Studio Community 2019

（四）源程序

```
#include<stdio.h>

void T(int x[5][5],int y[5][5], int n) //该函数用于实现矩阵转置并打印
{
    printf("矩阵A的转置矩阵为：\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            y[j][i] = x[i][j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%4d", y[i][j]);
        }
        printf("\n");
    }
}
void Plus(int x[5][5], int y[5][5], int z[5][5],  int n) //该函数用于实现矩阵A+B并打印
{
    printf("矩阵A+B为：\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            z[i][j] = x[i][j] + y[i][j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%4d", z[i][j]);
        }
        printf("\n");
    }
}
void Minus(int x[5][5], int y[5][5], int z[5][5], int n) //该函数用于实现矩阵A-B并打印
{
```

```
    printf("矩阵A-B为：\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            z[i][j] = x[i][j] - y[i][j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%4d", z[i][j]);
        }
        printf("\n");
    }
}
void Times(int x[5][5], int y[5][5], int z[5][5], int n) //该函数用于实现矩阵A×B并打印
{
    printf("矩阵A×B为：\n");
    for (int i = 0; i < n; i++)     //赋值
    {
        for (int j = 0; j < n; j++)
        {
            z[i][j] = 0;
            for (int k = 0; k < n; k++)
            {
                z[i][j] += x[i][k] * y[k][j];
            }
        }
    }
    for (int i = 0; i < n; i++)     //打印
    {
        for (int j = 0; j < n; j++)
        {
            printf("%4d", z[i][j]);
        }
        printf("\n");
    }
}
void Exponent2(int x[5][5], int z[5][5], int n) //该函数用于实现矩阵A^2并打印
{
    printf("矩阵A^2为：\n");
    for (int i = 0; i < n; i++)     //赋值
```

```
    {
        for (int j = 0; j < n; j++)
        {
            z[i][j] = 0;
            for (int k = 0; k < n; k++)
            {
                z[i][j] += x[i][k] * x[k][j];
            }
        }
    }
    for (int i = 0; i < n; i++)     //打印
    {
        for (int j = 0; j < n; j++)
        {
            printf("%4d", z[i][j]);
        }
        printf("\n");
    }
}
void Exponent3(int x[5][5], int y[5][5], int z[5][5], int n) //该函数用于实现矩阵A^3并打印
{
    printf("矩阵A^3为：\n");
    for (int i = 0; i < n; i++)     //赋值
    {
        for (int j = 0; j < n; j++)
        {
            z[i][j] = 0;
            for (int k = 0; k < n; k++)
            {
                z[i][j] += y[i][k] * x[k][j];
            }
        }
    }
    for (int i = 0; i < n; i++)     //打印
    {
        for (int j = 0; j < n; j++)
        {
            printf("%4d", z[i][j]);
        }
        printf("\n");
    }
}
int main()
```

```
{
    int A[5][5] = { 0 };
    int B[5][5] = { 0 };
    int n,i,j;
    int _T[5][5];     //转置
    int _Plus[5][5];     //相加
    int _Minus[5][5];     //相减
    int _Times[5][5];     //相乘
    int _Exponent2[5][5];     //幂运算
    int _Exponent3[5][5];
    printf("请输入方阵A、B的阶数n（n范围为1-5）：\n");
    scanf_s("%d", &n);
    printf("请输入矩阵A：\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf_s("%d", &A[i][j]);
        }
        getchar("\n");
    }
    printf("请输入矩阵B：\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf_s("%d", &B[i][j]);
        }
        getchar("\n");
    }

    T(A,_T, n);
    Plus(A,B,_Plus,n);
    Minus(A, B, _Minus, n);
    Times(A, B, _Times, n);
    Exponent2(A, _Exponent2, n);
    Exponent3(A, _Exponent2, _Exponent3, n);
    return 0;
}
```

（五）实验步骤与调试

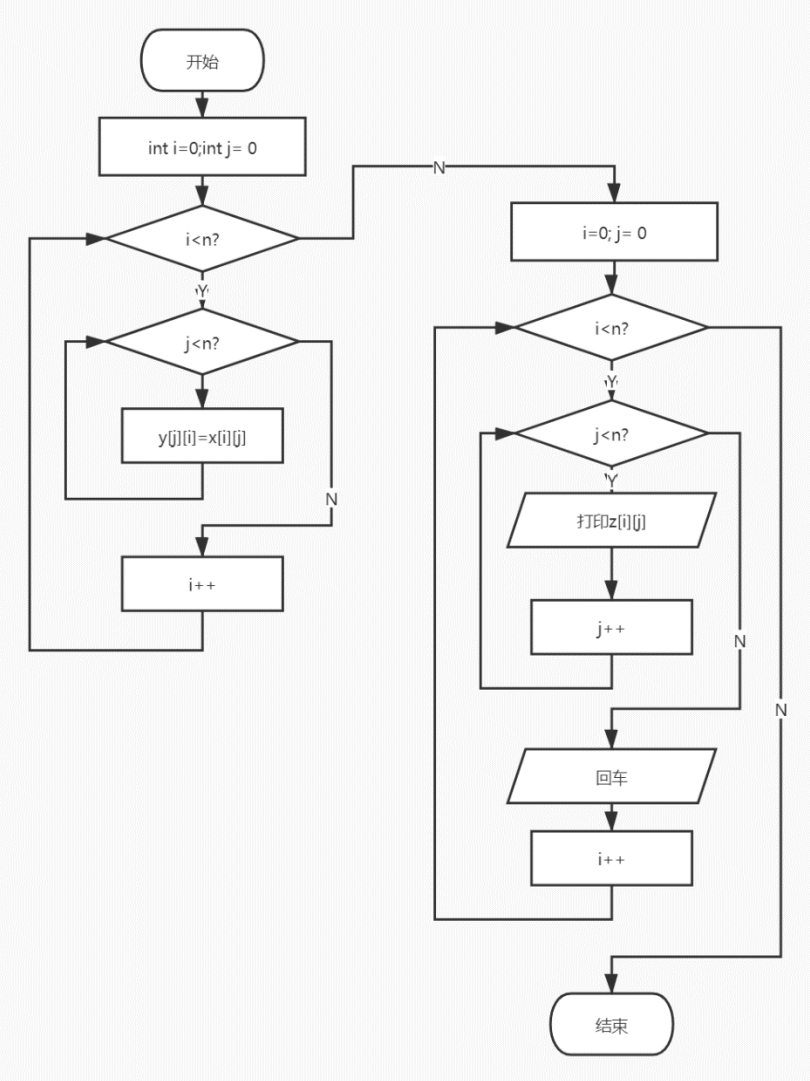
实验步骤：

1. 对实验进行基本构思，绘制流程图；
2. 启动 Visual Studio，创建新项目。将源程序写在新项目中；
3. 利用“本地 Windows 调试器”进行调试；
4. 进行多次调试并修正，直至得出理想结果。

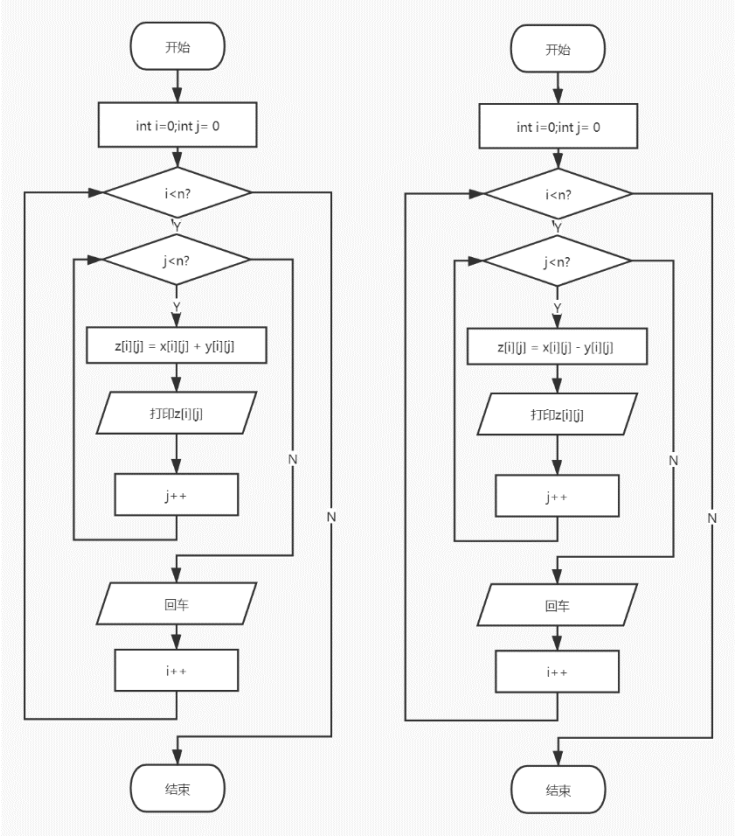
实验思路：

1. 因不熟悉函数封装的编写方式，决定使用先将各个矩阵计算的代码写在 main 函数中，再将相应代码提取的方式。
2. 各个矩阵操作思路：

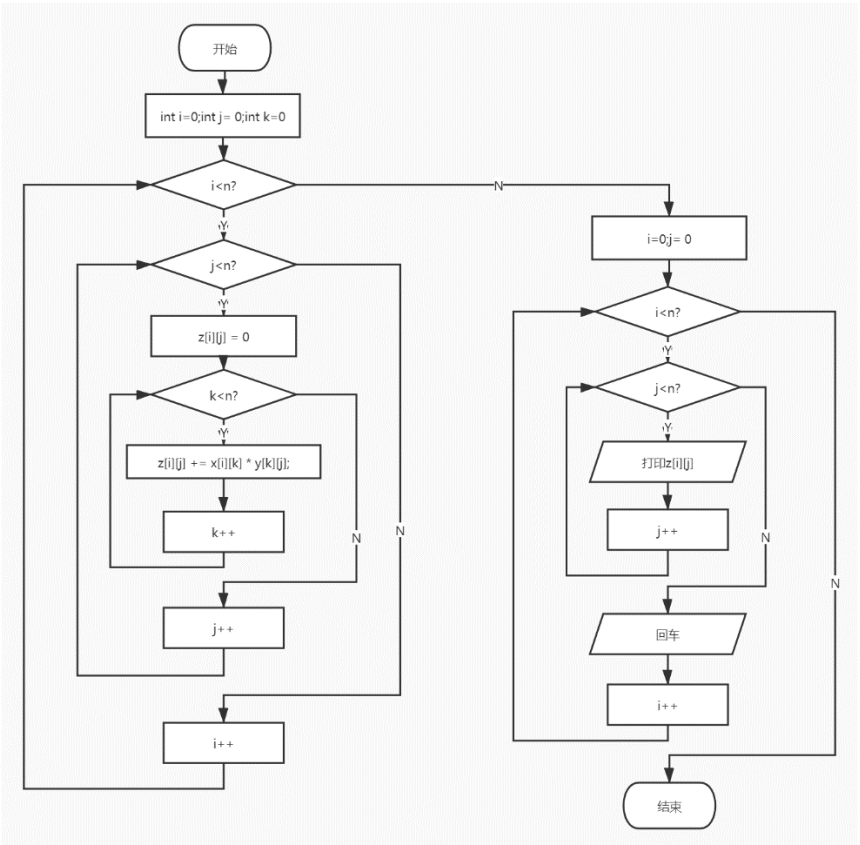
   【转置】



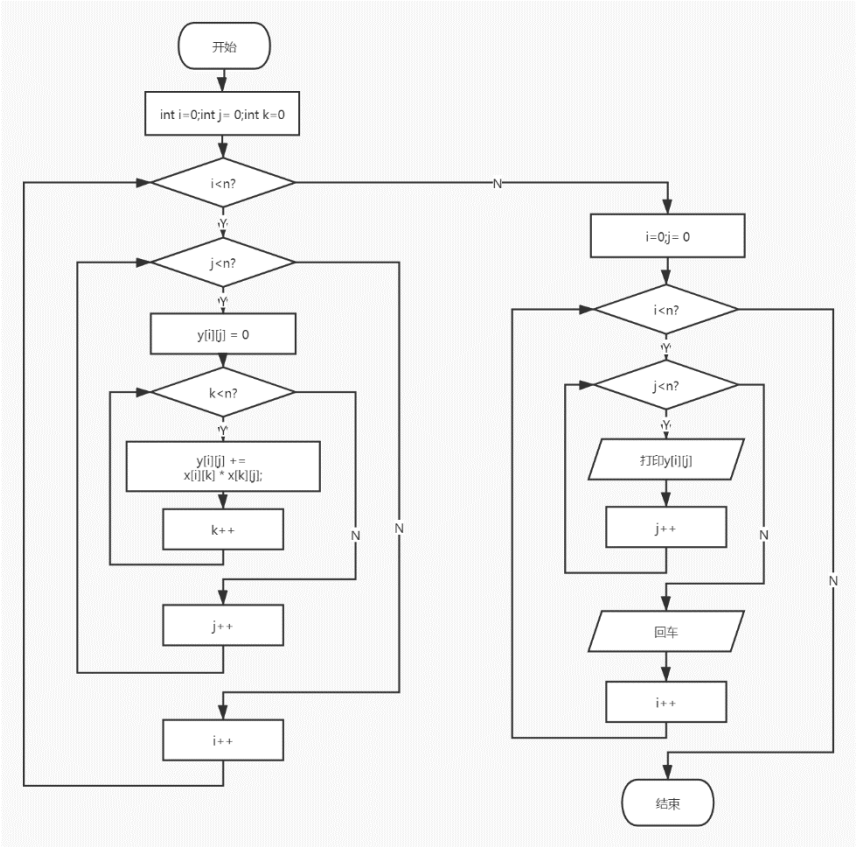
【相加、相减】


开始
int i=0;int j= 0
i<n?
Y
j<n?
Y
z[i][j] = x[i][j] + y[i][j]
打印z[i][j]
j++
N
回车
i++
N
结束
开始
int i=0;int j= 0
i<n?
Y
j<n?
Y
z[i][j] = x[i][j] - y[i][j]
打印z[i][j]
j++
N
回车
i++
N
结束

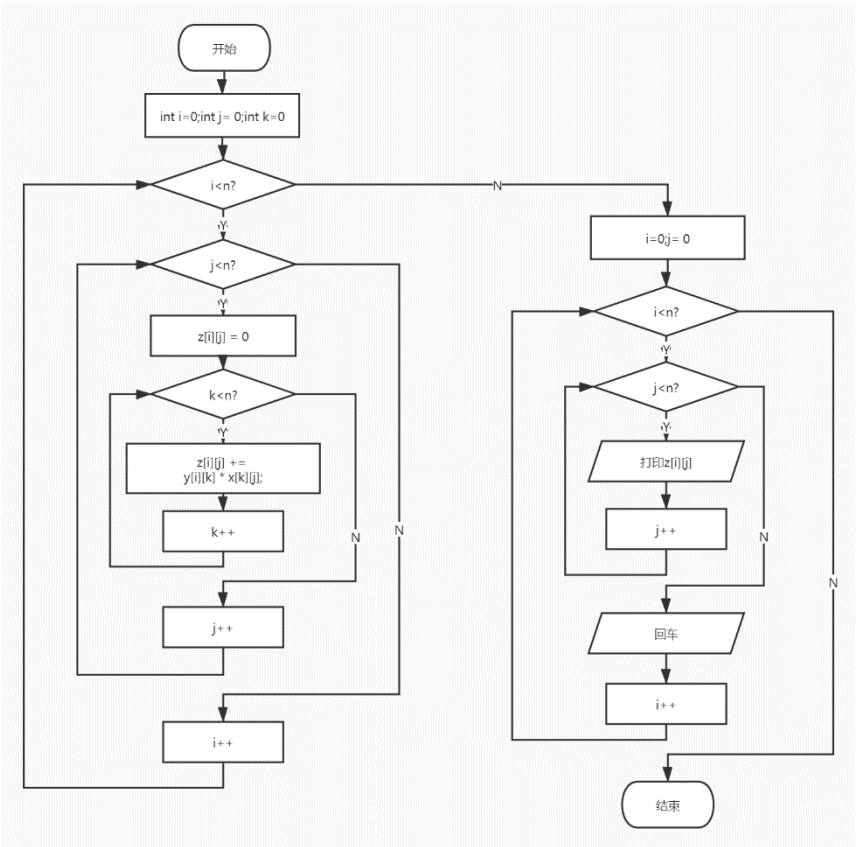
【相乘】


开始
int i=0;int j= 0;int k=0
i<n?
N
Y
j<n?
Y
z[i][j] = 0
k<n?
Y
z[i][j] += x[i][k] * y[k][j];
k++
N
j++
N
i++
i=0;j= 0
i<n?
Y
j<n?
Y
打印z[i][j]
j++
N
回车
i++
N
结束

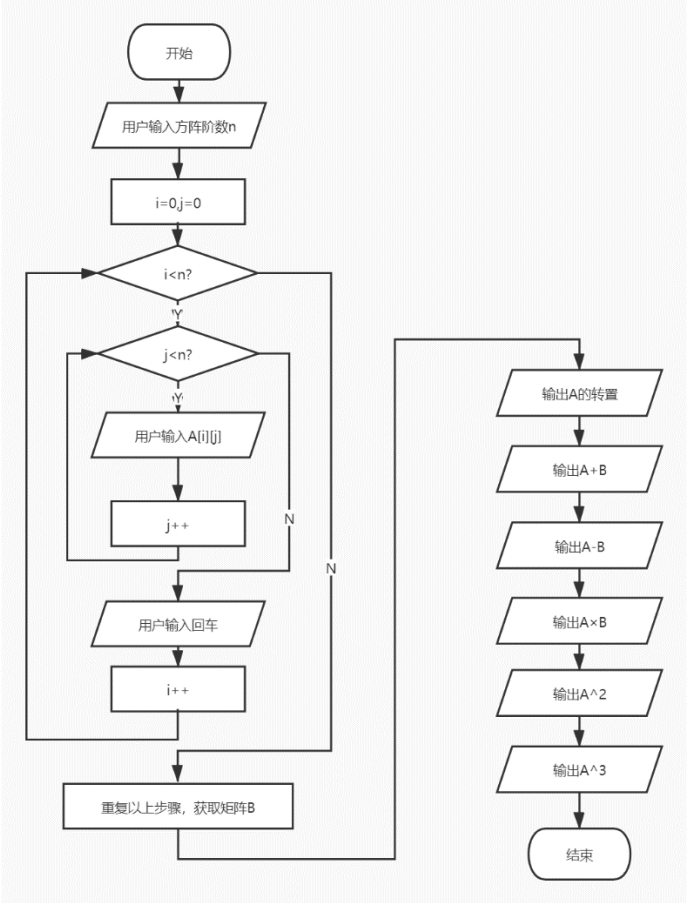
【2 次幂】


开始
int i=0;int j= 0;int k=0
i<n?
N
Y
j<n?
y[i][j] = 0
k<n?
y[i][j] +=
x[i][k] * x[k][j];
k++
j++
i++
i=0;j= 0
i<n?
j<n?
打印y[i][j]
j++
回车
i++
结束

【3 次幂】

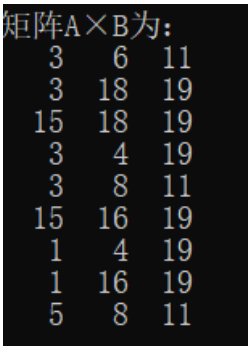

开始
int i=0;int j= 0;int k=0
i<n?
N
Y
j<n?
z[i][j] = 0
k<n?
z[i][j] +=
y[i][k] * x[k][j];
k++
j++
i++
i=0;j= 0
i<n?
j<n?
打印z[i][j]
j++
回车
i++
结束

【主程序】



调试：

1. 最初定义了 AB 均为 3×4 的矩阵，忽略了矩阵相乘需要的条件。虽然程序能够正常运行，但实际不存在这样的运算。回顾矩阵运算条件，若矩阵 A 与矩阵 B 能相乘，矩阵 A 的行数必须与矩阵 B 的列数相等。考虑到程序同时要求打印 A^n，矩阵 A 必为方阵。同时矩阵 A、B 可以进行加法减法运算，其行数列数必定相同。所以矩阵 A、B 必为同阶方阵，重新将矩阵 A、B 定义为 3×3 的矩阵。
2. 最初矩阵 A×B 的结果为：



明显与矩阵相乘结果不符。检查相应代码后发现：

```
62      for (i = 0; i < 3; i++)
63      {
64          for (j = 0; j < 3; j++)
65          {
66              Times[i][j] = 0;
67              for (k = 0; k < 3; k++)
68              {
69                  Times[i][j] += A[i][k] * B[k][j];
70                  printf("%4d", Times[i][j]);
71              }
72              printf("\n");
73          }
74      }
```

该段代码在给矩阵 Times 赋值的过程中打印了该矩阵，打印时分开打印了其中的计算过程，即没有完成计算就开始打印。删去 70、72 行的语句，在赋值的循环嵌套后再加入打印矩阵 A×B 结果的循环嵌套，解决了该问题。

```
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 3; j++)
    {
        Times[i][j] = 0;
        for (k = 0; k < 3; k++)
        {
            Times[i][j] += A[i][k] * B[k][j];
        }
    }
}
```

```
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 3; j++)
    {
        printf("%4d", Times[i][j]);
    }
    printf("\n");
}
```

```
矩阵A×B为:
  11   19   19
  19   11   19
  19   19   11
```

3. 程序可以正常运行，且符合实验的运算要求。说明计算代码正确。

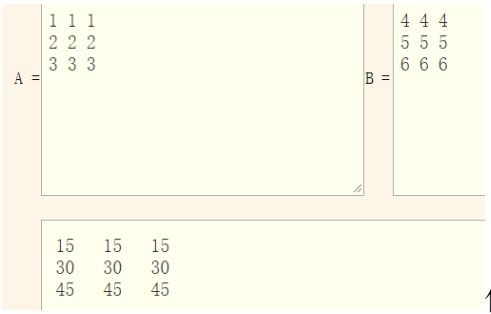
矩阵运算的函数封装

```
10      int Minus[3][3];    //矩阵相减
11      int Times[3][3];    //矩阵相乘
12      int Exponent2[3][3];    //幂运算
13      int Exponent3[3][3];
14
15      printf("矩阵A为: \n");
16      for (i = 0; i < 3; i++){ ... }
25      printf("矩阵B为: \n");
26      for (i = 0; i < 3; i++){ ... }
34      printf("矩阵A的转置矩阵为: \n");
35      for (i = 0; i < 3; i++){ ... }
43      printf("矩阵A+B为: \n");
44      for (i = 0; i < 3; i++){ ... }
53      printf("矩阵A-B为: \n");
54      for (i = 0; i < 3; i++){ ... }
63      printf("矩阵A×B为: \n");
64      for (i = 0; i < 3; i++){ ... }
75      for (i = 0; i < 3; i++){ ... }
83      printf("矩阵A^2为: \n");
84      for (i = 0; i < 3; i++){ ... }
95      for (i = 0; i < 3; i++){ ... }
103     printf("矩阵A^3为: \n");
104     for (i = 0; i < 3; i++){ ... }
115     for (i = 0; i < 3; i++){ ... }
123
```

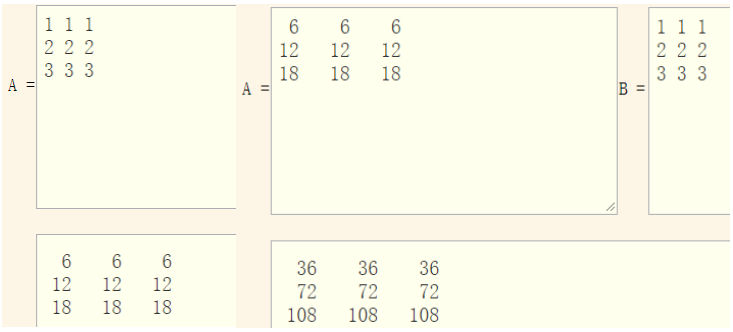
% 未找到相关问题

选择Microsoft Visual Studi

```
矩阵A为:
   1    1    1
   2    2    2
   3    3    3
矩阵B为:
   4    4    4
   5    5    5
   6    6    6
矩阵A的转置矩阵为:
   1    2    3
   1    2    3
   1    2    3
矩阵A+B为:
   5    5    5
   7    7    7
   9    9    9
矩阵A-B为:
  -3   -3   -3
  -3   -3   -3
  -3   -3   -3
矩阵A×B为:
  15   15   15
  30   30   30
  45   45   45
矩阵A^2为:
   6    6    6
  12   12   12
  18   18   18
矩阵A^3为:
  36   36   36
  72   72   72
 108  108  108
```



借助计算器验算得 A×B 结果无误。

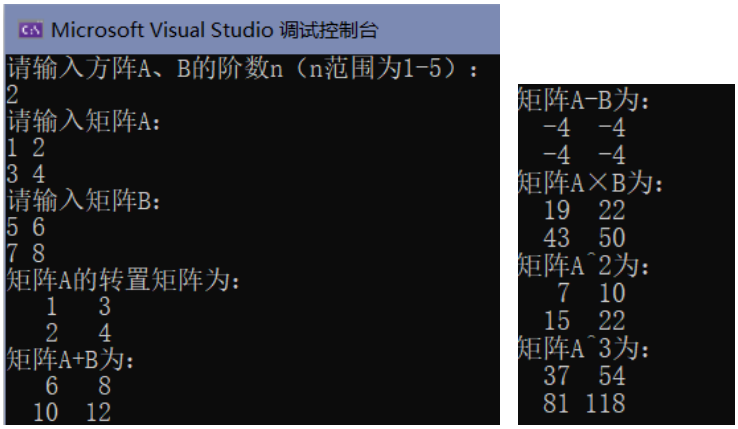


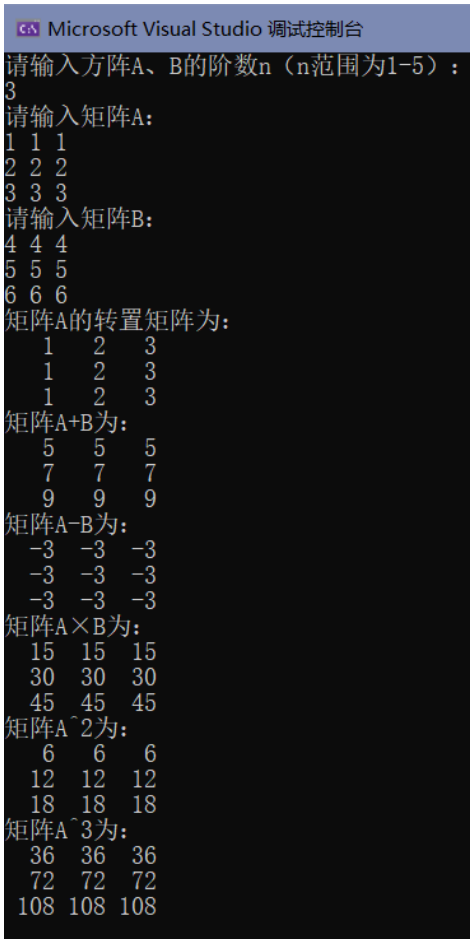
A^2 以及 A^3 结果无误。为使程序功能更加丰富，考虑 A、B 不由程序自定义，而是由用户输入。同时，保证各个矩阵计算过程代码无误后将相应代码提取，用函数封装各个操作。

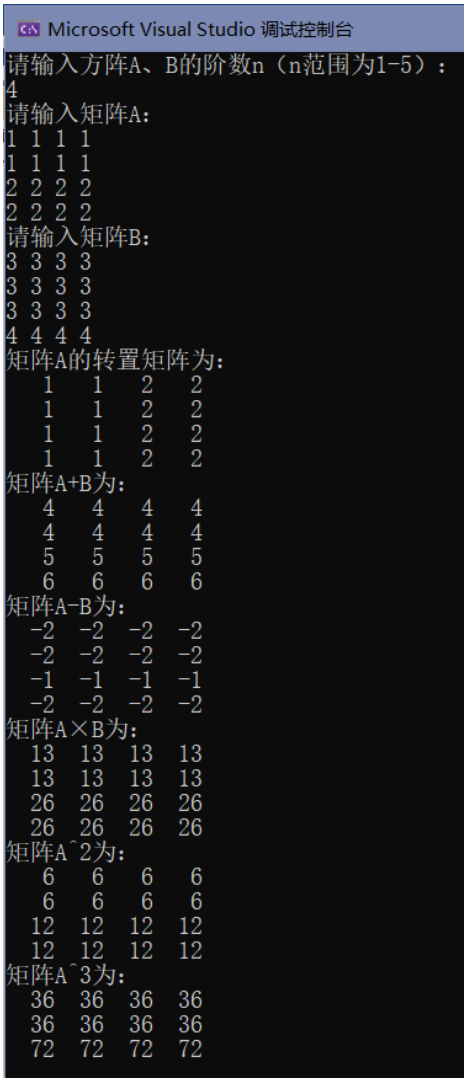
```
#include<stdio.h>

void T(int x[5][5],int y[5][5], int n){ ... }
void Plus(int x[5][5], int y[5][5], int z[5][5],  int n){ ... }
void Minus(int x[5][5], int y[5][5], int z[5][5], int n){ ... }
void Times(int x[5][5], int y[5][5], int z[5][5], int n){ ... }
void Exponent2(int x[5][5], int z[5][5], int n){ ... }
void Exponent3(int x[5][5], int y[5][5], int z[5][5], int n){ ... }
int main()
{
```

（六）实验结果与分析

实验结果：







1. 程序可按实验要求运行；
2. 程序情况基本符合理想效果；
3. 调试过程中出现的问题均已解决。

分析：

1. 事先获取用户输入方阵的阶数，可以确定用于循环流程控制的条件，便于程序的编写，可使程序的功能更加丰富。
2. 用户输入矩阵时每输入一行，用 getchar()函数将用户输入的回车存到缓冲区，使用户可以输入矩阵原有格式，输入时可以更加直观。
3. 查阅线性代数相关知识，

$$(AB)_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{ip} b_{pj}$$

如下所示：

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

$$C = AB = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + a_{1,3}b_{3,1}, & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} + a_{1,3}b_{3,2} \\ \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} + a_{2,3}b_{3,1}, & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} + a_{2,3}b_{3,2} \end{bmatrix}$$

可以看出，c[i][j]=a[i][1]b[1][j]+ a[i][2]b[2][j]+…+ a[i][n]b[n][j]，其中 n 为矩阵 b 的列数。因此可以在计算矩阵 c 中某个数 c[i][j]时使用循环语句。由此在计算矩阵相乘时使用三重循环嵌套。进行矩阵幂运算同理。

4. 由线性代数相关知识，矩阵幂运算中 A^2=A×A，A^3= A^2×A。因此在编写代码时计算 A^2 时可以直接将 A×B 中的 B 改为 A：

```
{
    Times[i][j] += A[i][k] * B[k][j];
}
```

```
{
    Exponent2[i][j] += A[i][k] * A[k][j];
}
```

计算 A^3 时同理。

```
{
    Exponent3[i][j] += Exponent2[i][k] * A[k][j];
}
```