**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 成绩评定

实验项目名称 指导教师 张鑫源

实验项目编号 实验项目类型 实验地点

学生姓名 张景曦 学号 2019051098

学院 智科 系 专业 信息安全

实验时间 2020 年 月 日 午～ 5 月 12 午 温度 ℃湿度

1. **实验目的**
2. **实验内容和要求**
3. **主要仪器设备**

**仪器：**计算机

**实验环境：vs**

1. **源程序**

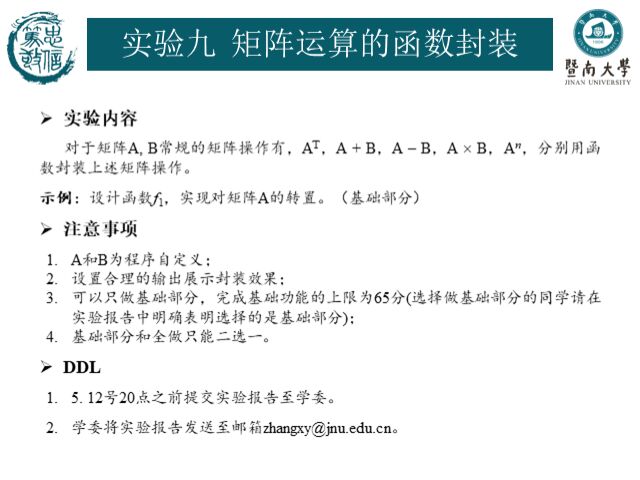
写出程序的源程序

1. **实验步骤与调试**

主要是写出对源程序的语法错误以及逻辑错误的进行发现、修正以及调试的步骤。

1. **实验结果与分析**

写出最终逻辑正确的程序在各种实验数据下的实验结果以及进行简单的结果分析。



源代码：

#include <stdio.h>

int f1(int a[3][3]) //定义函数实现矩阵A的转置

{

int i, j;

printf("输出矩阵A的转置：\n");

for (i = 0;i < 3;i++)

{

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%d\n", a[j][i]);

else printf("%d ", a[j][i]);

}

}

printf("\n");

return 0;

}

int f2(int a[3][3],int b[3][3]) //定义函数实现矩阵A与矩阵B的相加

{

int i, j;

printf("输出矩阵A+B：\n");

for(i=0;i<3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%d\n", a[i][j] + b[i][j]);

else

printf("%d ", a[i][j] + b[i][j]);

}

printf("\n");

return 0;

}

int f3(int a[3][3], int b[3][3]) //定义函数实现矩阵A与矩阵B的相减

{

int i, j;

printf("输出矩阵A-B：\n");

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-2d\n", a[i][j] - b[i][j]);

else

printf("%-2d ", a[i][j] - b[i][j]);

//通过%-2d左对齐防止因为出现负数导致的不美观

}

printf("\n");

return 0;

}

int f4(int a[3][3], int b[3][3]) //定义函数实现矩阵A与矩阵B的相乘

{

int i, j;

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

{

printf("%-2d\n", a[i][0] \* b[0][j] + a[i][1] \* b[1][j] + a[i][2] \* b[2][j]);

}

else

{

printf("%-2d ", a[i][0] \* b[0][j] + a[i][1] \* b[1][j] + a[i][2] \* b[2][j]);

}

}

printf("\n");

return 0;

}

int f5(int a[3][3], int n) //定义函数实现矩阵A的n次幂运算

{

int i, j, k;

int c[3][3], d[3][3] = { 0 };

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

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

if(n==1)

for(i=0;i<3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-d\n", a[i][j]);

else printf("%-d ", a[i][j]);

}

else

{

for (k = 1;k < n;k++)

{

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

d[i][j] = c[i][0] \* a[0][j] + c[i][1] \* a[1][j] + c[i][2] \* a[2][j];

}

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

c[i][j] = d[i][j];

}

for (i = 0;i < 3;i++)

{

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-d \n", c[i][j]);

else printf("%-d ", c[i][j]);

}

}

}

printf("\n");

return 0;

}

int main()

{

int i, j,n, a[3][3],b[3][3];

printf("请输入矩阵A：\n");

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

scanf("%d", &a[i][j]);

}

printf("请输入矩阵B：\n");

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

scanf("%d", &b[i][j]);

}

f1(a);

f2(a, b);

f3(a, b);

printf("输出矩阵A和矩阵B的乘积：\n");

f4(a, b);

printf("请输入矩阵A的幂数n:");

scanf("%d", &n);

f5(a, n);

return 0;

}

**以上为最终的代码。**

**下面是未修改的代码，存在错误所以进行了修改：**

#include <stdio.h>

int f1(int a[3][3]) //定义函数实现矩阵A的转置

{

int i, j;

printf("输出矩阵A的转置：\n");

for (i = 0;i < 3;i++)

{

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%d\n", a[j][i]);

else printf("%d ", a[j][i]);

}

}

printf("\n");

return 0;

}

int f2(int a[3][3],int b[3][3]) //定义函数实现矩阵A与矩阵B的相加

{

int i, j;

printf("输出矩阵A+B：\n");

for(i=0;i<3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%d\n", a[i][j] + b[i][j]);

else

printf("%d ", a[i][j] + b[i][j]);

}

printf("\n");

return 0;

}

int f3(int a[3][3], int b[3][3]) //定义函数实现矩阵A与矩阵B的相减

{

int i, j;

printf("输出矩阵A-B：\n");

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-2d\n", a[i][j] - b[i][j]);

else

printf("%-2d ", a[i][j] - b[i][j]);

//通过%-2d左对齐防止因为出现负数导致的不美观

}

printf("\n");

return 0;

}

int f4(int a[3][3], int b[3][3]) //定义函数实现矩阵A与矩阵B的相乘

{

int i, j;

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

{

printf("%-2d\n", a[i][0] \* b[0][j] + a[i][1] \* b[1][j] + a[i][2] \* b[2][j]);

}

else

{

printf("%-2d ", a[i][0] \* b[0][j] + a[i][1] \* b[1][j] + a[i][2] \* b[2][j]);

}

}

printf("\n");

return 0;

}

int f5(int a[3][3], int n) //定义函数实现矩阵A的n次幂运算

{

int i, j, k;

int c[3][3];

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

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

if(n==1)

for(i=0;i<3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-d\n", c[i][j]);

else printf("%-d ", c[i][j]);

}

else

{

for (k = 1;k < n;k++)

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

c[i][j] = c[i][0] \* a[0][j] + c[i][1] \* a[1][j] + c[i][2] \* a[2][j];

}

for (i = 0;i < 3;i++)

{

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-d \n", c[i][j]);

else printf("%-d ", c[i][j]);

}

}

}

printf("\n");

return 0;

}

int main()

{

int i, j,n, a[3][3],b[3][3];

printf("请输入矩阵A：\n");

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

scanf("%d", &a[i][j]);

}

printf("请输入矩阵B：\n");

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

scanf("%d", &b[i][j]);

}

f1(a);

f2(a, b);

f3(a, b);

printf("输出矩阵A和矩阵B的乘积：\n");

f4(a, b);

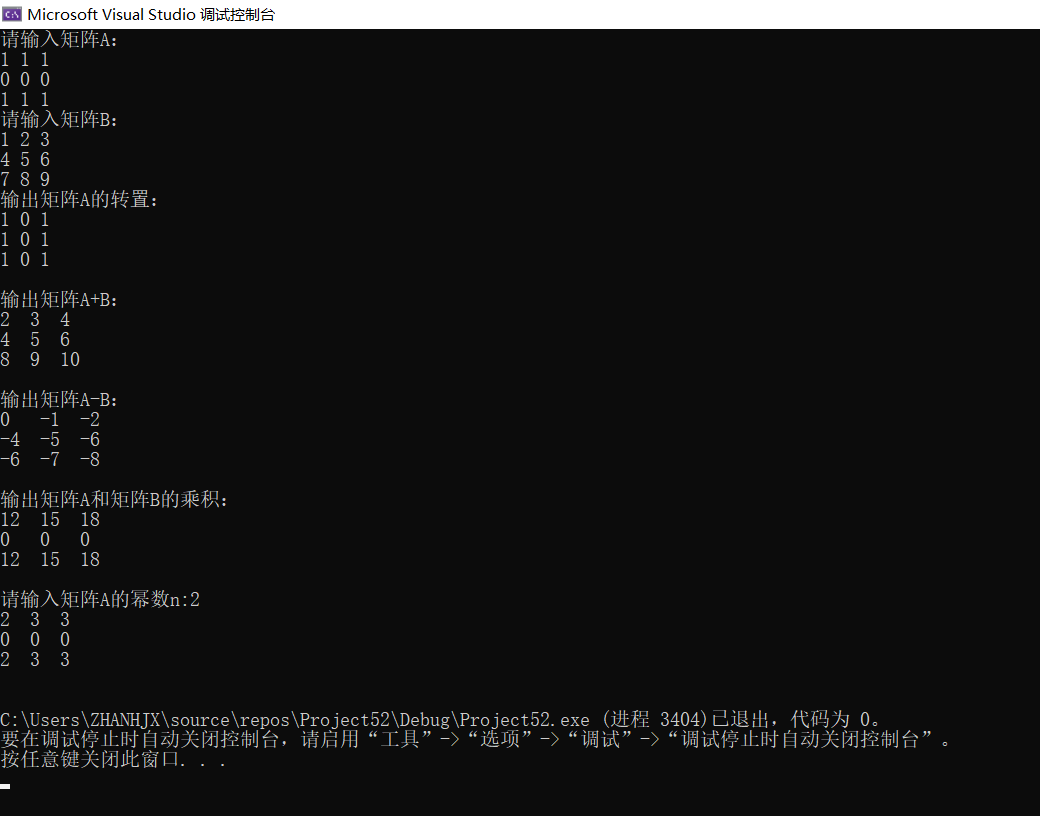
printf("请输入矩阵A的幂数n:");

scanf("%d", &n);

f5(a, n);

return 0;

}



第一次尝试发现最后求幂的结果是错误的。

大致是因为

for (k = 1;k < n;k++)

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

c[i][j] = c[i][0] \* a[0][j] + c[i][1] \* a[1][j] + c[i][2] \* a[2][j];

}

出现了问题：当i=0,j=0时c[0][0]得到了一个新值2，但下一步求i[0][1]时仍要使用到i[0][0](本应该是数值1)但结果却是以2来计算的。

做出如下改进：

int f5(int a[3][3], int n) //定义函数实现矩阵A的n次幂运算

{

int i, j, k;

int c[3][3], d[3][3] = { 0 };

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

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

if(n==1)

for(i=0;i<3;i++)

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-d\n", a[i][j]);

else printf("%-d ", a[i][j]);

}

else

{

for (k = 1;k < n;k++)

{

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

{

d[i][j] = c[i][0] \* a[0][j] + c[i][1] \* a[1][j] + c[i][2] \* a[2][j];

}

for (i = 0;i < 3;i++)

for (j = 0;j < 3;j++)

c[i][j] = d[i][j];

}

for (i = 0;i < 3;i++)

{

for (j = 0;j < 3;j++)

{

if (j == 2)

printf("%-d \n", c[i][j]);

else printf("%-d ", c[i][j]);

}

}

}

printf("\n");

return 0;

}

重新定义了一个空的二维数组来储存每次做幂之后的结果，使得问题得到了解决。

