**图像变换算法实现I**

1. **实验内容**

【问题描述】

输入一幅图（bmp文件或者自建图形编辑系统），将图进行平移变换（水平距离和垂直距离，或移动距离和移动角度）以及旋转变换（指定旋转角度、旋转中心为画面中心）。

【基本要求】

（1）输入：一幅图，以及用户需求。

（2）输出：变换后的图。

【实现提示】

（1）bmp文件转换一个矩阵存储（需自己查资料实现）

（2）平移和旋转变换都是矩阵变换应用

**二、数据结构设计**

无

**三、算法设计**

#include<math.h>

#include<iomanip>

#include<stdlib.h>

#include<windows.h>

#include<stdio.h>

#include<stdlib.h>

#include<iostream>

#include<fstream>

using namespace std;

#define PI 3.1415926

unsigned char \*pBmpBuf;//读入图像数据的指针

int bmpWidth;//图像的宽

int bmpHeight;//图像的高

RGBQUAD \*pColorTable;//颜色表指针

int biBitCount;//图像类型，每像素位数

//读图像的位图数据、宽、高、颜色表及每像素位数等数据进内存，存放在相应的全局变量中

bool readBmp(char \*bmpName)

{

FILE \*fp=fopen(bmpName,"rb");//二进制读方式打开指定的图像文件

if(fp==0)

return 0;

//跳过位图文件头结构BITMAPFILEHEADER

fseek(fp, sizeof(BITMAPFILEHEADER),0);

//定义位图信息头结构变量，读取位图信息头进内存，存放在变量head中

BITMAPINFOHEADER head;

fread(&head, sizeof(BITMAPINFOHEADER), 1,fp); //获取图像宽、高、每像素所占位数等信息

bmpWidth = head.biWidth;

bmpHeight = head.biHeight;

biBitCount = head.biBitCount;//定义变量，计算图像每行像素所占的字节数（必须是4的倍数）

int lineByte=(bmpWidth \* biBitCount/8+3)/4\*4;//灰度图像有颜色表，且颜色表表项为256

if(biBitCount==8)

{

//申请颜色表所需要的空间，读颜色表进内存

pColorTable=new RGBQUAD[256];

fread(pColorTable,sizeof(RGBQUAD),256,fp);

}

//申请位图数据所需要的空间，读位图数据进内存

pBmpBuf=new unsigned char[lineByte \* bmpHeight];

fread(pBmpBuf,1,lineByte \* bmpHeight,fp);

fclose(fp);//关闭文件

return 1;//读取文件成功

}

//给定一个图像位图数据、宽、高、颜色表指针及每像素所占的位数等信息,将其写到指定文件中

bool saveBmp(char \*bmpName, unsigned char \*imgBuf, int width, int height, int biBitCount, RGBQUAD \*pColorTable)

{

//如果位图数据指针为0，则没有数据传入，函数返回

if(!imgBuf)

return 0;

//颜色表大小，以字节为单位，灰度图像颜色表为1024字节，彩色图像颜色表大小为0

int colorTablesize=0;

if(biBitCount==8)

colorTablesize=1024;

//待存储图像数据每行字节数为4的倍数

int lineByte=(width \* biBitCount/8+3)/4\*4;

//以二进制写的方式打开文件

FILE \*fp=fopen(bmpName,"wb");

if(fp==0)

return 0;

//申请位图文件头结构变量，填写文件头信息

BITMAPFILEHEADER fileHead;

fileHead.bfType = 0x4D42;//bmp类型

//bfSize是图像文件4个组成部分之和

fileHead.bfSize= sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + colorTablesize + lineByte\*height;

fileHead.bfReserved1 = 0;

fileHead.bfReserved2 = 0;

//bfOffBits是图像文件前3个部分所需空间之和

fileHead.bfOffBits=54+colorTablesize;

//写文件头进文件

fwrite(&fileHead, sizeof(BITMAPFILEHEADER),1, fp);

//申请位图信息头结构变量，填写信息头信息

BITMAPINFOHEADER head;

head.biBitCount=biBitCount;

head.biClrImportant=0;

head.biClrUsed=0;

head.biCompression=0;

head.biHeight=height;

head.biPlanes=1;

head.biSize=40;

head.biSizeImage=lineByte\*height;

head.biWidth=width;

head.biXPelsPerMeter=0;

head.biYPelsPerMeter=0;

//写位图信息头进内存

fwrite(&head, sizeof(BITMAPINFOHEADER),1, fp);

//如果灰度图像，有颜色表，写入文件

if(biBitCount==8)

fwrite(pColorTable, sizeof(RGBQUAD),256, fp);

//写位图数据进文件

fwrite(imgBuf, height\*lineByte, 1, fp);

//关闭文件

fclose(fp);

return 1;

}

//以下为像素的读取函数

void doIt()

{

//读入指定BMP文件进内存

char readPath[]="9ebb0c3dbfdfb9b483695a772441e66d.BMP";

readBmp(readPath);

//输出图像的信息

// cout<<"width="<<bmpWidth<<" height="<<bmpHeight<<" biBitCount="<<biBitCount<<endl;

//循环变量，图像的坐标

//每行字节数

/\*CWnd \*pWnd=GetDlgItem(IDC\_PICTURE);//获得pictrue控件窗口的句柄

CRect rect;

pWnd->GetClientRect(&rect);//获得pictrue控件所在的矩形区域

CDC \*pDC=pWnd->GetDC();//获得pictrue控件的DC

//显示图片

pDC->SetStretchBltMode(COLORONCOLOR);

StretchDIBits(pDC->GetSafeHdc(),0,0,rect.Width(),rect.Height(),0,0,bmpWidth,bmpHeight,pBmpData,pBmpInfo,DIB\_RGB\_COLORS,SRCCOPY);\*/

cout<<" ";

for(int i=0;i<30;i++)

printf("===");

cout<<endl;

cout<<" || ";

int lineByte=(bmpWidth\*biBitCount/8+3)/4\*4;

//循环变量，针对彩色图像，遍历每像素的三个分量

int m=0,n=0,count\_xiang\_su=0,choose;

cout<<" 请选择对图片进行的操作: 1缺失平移变换 2旋转变换 3不缺失平移变换 ||\n";

cin>>choose;

//将图像左下角1/4部分置成黑色

/\* ofstream outfile("图像像素.txt",ios::in|ios::trunc);

if(biBitCount==8) //对于灰度图像

{

//------------------------------------------------------------------------------------

//以下完成图像的分割成8\*8小单元，并把像素值存储到指定文本中。由于BMP图像的像素数据是从

//左下角：由左往右，由上往下逐行扫描的

int L1=0;

int hang=63;

int lie=0;

//int L2=0;

//int fen\_ge=8;

for(int fen\_ge\_hang=0;fen\_ge\_hang<8;fen\_ge\_hang++)//64\*64矩阵行循环

{

for(int fen\_ge\_lie=0;fen\_ge\_lie<8;fen\_ge\_lie++)//64\*64列矩阵循环

{

//--------------------------------------------

for(L1=hang;L1>hang-8;L1--)//8\*8矩阵行

{

for(int L2=lie;L2<lie+8;L2++)//8\*8矩阵列

{

m=\*(pBmpBuf+L1\*lineByte+L2);

outfile<<m<<" ";

count\_xiang\_su++;

if(count\_xiang\_su%8==0)//每8\*8矩阵读入文本文件

{

outfile<<endl;

}

}

}

//---------------------------------------------

hang=63-fen\_ge\_hang\*8;//64\*64矩阵行变换

lie+=8;//64\*64矩阵列变换

//该一行（64）由8个8\*8矩阵的行组成

}

hang-=8;//64\*64矩阵的列变换

lie=0;//64\*64juzhen

}

}

//double xiang\_su[2048];

//ofstream outfile("xiang\_su\_zhi.txt",ios::in|ios::trunc);

if(!outfile)

{

cout<<"open error!"<<endl;

exit(1);

} \*/

if(choose==2)

{

int i,j,x,y,x2,y2,intex,intey,original\_point\_A,original\_point\_B,original\_point\_C,original\_point\_D,m,n;

double x1,y1,decix,deciy;

double p;

cout<<"请输入旋转角度:";

cin>>p;

if ((int)p % 360 == 270){

p ++;

}

p=(double)(p\*PI/180);

int newwidth,newheigh;

newwidth=abs(bmpWidth \* cos(p) + bmpHeight \* sin(p));

newheigh=abs(bmpWidth \* sin(p) + bmpHeight \* cos(p));

//cout<<bmpWidth<<" "<<bmpHeight<<endl<<newwidth<<" "<<newheigh<<endl;

int newlineByte=(newwidth \* biBitCount/8+3)/4\*4;

unsigned char \*newpBmpBuf;

newpBmpBuf=new unsigned char[newlineByte \* newheigh];

x=newwidth/2;

y=newheigh/2;

x2=bmpWidth/2;

y2=bmpHeight/2;

//cout<<x<<" "<<y<<" "<<x2<<" "<<y2<<endl;

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

{

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

{

y1=(i-y)\*cos(-p)+(j-x)\*sin(-p);

x1=(j-x)\*cos(-p)-(i-y)\*sin(-p);

if(abs(x1)+x2<bmpWidth&&abs(y1)+y2<bmpHeight)

{

intex=x1+x2;

intey=y1+y2;

/\*decix=x1-intex;

deciy=y1-intey;

original\_point\_A = intey \* lineByte + intex \* 3;

original\_point\_B = intey \* lineByte + (intex + 1) \* 3;

original\_point\_C = (intey + 1) \* lineByte + intex \* 3;

original\_point\_D = (intey + 1) \* lineByte + (intex + 1)\*3;

if (intex == bmpWidth - 1){

// cout << "hhhhh" << endl;

original\_point\_B = original\_point\_A;

original\_point\_D = original\_point\_C;

}

if (intey == bmpHeight - 1){

original\_point\_C = original\_point\_A;

original\_point\_D = original\_point\_B;

}

for(int k=0;k<3;k++)

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=(1-deciy)\*(1-decix)\*\*(pBmpBuf+original\_point\_A+k)+(1-deciy)\*decix\*\*(pBmpBuf+original\_point\_C+k)+deciy\*(1-decix)\*\*(pBmpBuf+original\_point\_B+k)+deciy\*decix\*\*(pBmpBuf+original\_point\_D+k);

}\*/

for(int k=0;k<3;k++)

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=\*(pBmpBuf+intey \* lineByte + intex \* 3+k);

}

continue;

}

for(int k=0;k<3;k++)

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

char writePath[]="nvcpy.BMP";//图片处理后再存储

saveBmp(writePath, newpBmpBuf, newwidth, newheigh, biBitCount, pColorTable);

//清除缓冲区，pBmpBuf和pColorTable是全局变量，在文件读入时申请的空间

delete []pBmpBuf;

if(biBitCount==8)

delete []pColorTable;

cout<<"旋转图片完成!";

return;

}

else if(choose==3)

{

int yhei,hei,ywid,wid,newHeight,newWidth,newlineByte;

cout<<"请输入你想要平移高度(负数为下移，正数为上移):";

cin>>yhei;

hei=fabs(yhei);

cout<<"请输入你想要平移宽度(负数为左移，正数为右移):";

cin>>ywid;

unsigned char \*newpBmpBuf;

wid=fabs(ywid);

newHeight=bmpHeight+hei;

newWidth=bmpWidth+wid;

newlineByte=lineByte+wid\*3;

newpBmpBuf=new unsigned char[(lineByte+wid\*3) \* (bmpHeight+hei)];

int i,j,k;

if(yhei<=0)

{

for(i=0;i<newHeight-hei;i++)

{

if(ywid<=0)

{

for(j=0;j<newWidth-wid;j++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=\*(pBmpBuf+i\*lineByte+j\*3+k);

}

}

}

else

{

for(j=newWidth-1;j>=wid;j--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=\*(pBmpBuf+i\*lineByte+(j-wid)\*3+k);

}

}

}

}

for(i=newHeight-1;i>=newHeight-hei;i--)

{

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

{

for(int k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

}

if(ywid>0)

{

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

{

for(int i=0;i<newHeight-hei;i++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

}

}

else

{

for(j=newWidth-1;j>=newWidth-wid;j--)

{

for(int i=0;i<newHeight-hei;i++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

}

}

}

else

{

for(i=newHeight-1;i>=hei;i--)

{

if(ywid<=0)

{

for(j=0;j<newWidth-wid;j++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=\*(pBmpBuf+(i-hei)\*lineByte+j\*3+k);

}

}

}

else

{

for(j=newWidth-1;j>=wid;j--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=\*(pBmpBuf+(i-hei)\*lineByte+(j-wid)\*3+k);

}

}

}

}

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

{

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

{

for(int k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

}

if(ywid>0)

{

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

{

for(i=newHeight-1;i>=hei;i--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

}

}

else

{

for(j=newWidth-1;j>=newWidth-wid;j--)

{

for(i=newHeight-1;i>=hei;i--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(newpBmpBuf+i\*newlineByte+j\*3+k)=0;

}

}

}

}

}

/\* for(int i=0;i<bmpHeight;i++)

{

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

{

for(int k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

//\*(pBmpBuf+i\*lineByte+j\*3+k)-=40;

m=\*(pBmpBuf+i\*lineByte+j\*3+k);

outfile<<m<<" ";

count\_xiang\_su++;

if(count\_xiang\_su%8==0)

{

outfile<<endl;

}

//n++;

}

n++;

}

}

cout<<"总的像素个素为:"<<n<<endl;

cout<<"----------------------------------------------------"<<endl; \*/

char writePath[]="nvcpy.BMP";//图片处理后再存储

saveBmp(writePath, newpBmpBuf, newWidth, newHeight, biBitCount, pColorTable);

//清除缓冲区，pBmpBuf和pColorTable是全局变量，在文件读入时申请的空间

delete []pBmpBuf;

if(biBitCount==8)

delete []pColorTable;

cout<<"平移图片完成！";

return;

}

else if(choose==1)

{//彩色图像

int yhei,hei,ywid,wid;

cout<<"请输入你想要平移高度(负数为下移，正数为上移):";

cin>>yhei;

hei=fabs(yhei);

cout<<"请输入你想要平移宽度(负数为左移，正数为右移):";

cin>>ywid;

wid=fabs(ywid);

int i,j,k;

if(yhei<=0)

{

for(i=0;i<bmpHeight-hei;i++)

{

if(ywid<=0)

{

for(j=0;j<bmpWidth-wid;j++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=\*(pBmpBuf+(i+hei)\*lineByte+(j+wid)\*3+k);

}

}

}

else

{

for(j=bmpWidth-1;j>=wid;j--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=\*(pBmpBuf+(i+hei)\*lineByte+(j-wid)\*3+k);

}

}

}

}

for(i=bmpHeight-1;i>=bmpHeight-hei;i--)

{

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

{

for(int k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=0;

}

}

}

if(ywid>0)

{

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

{

for(int i=0;i<bmpHeight-hei;i++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=0;

}

}

}

}

else

{

for(j=bmpWidth-1;j>=bmpWidth-wid;j--)

{

for(int i=0;i<bmpHeight-hei;i++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=0;

}

}

}

}

}

else

{

for(i=bmpHeight-1;i>=hei;i--)

{

if(ywid<=0)

{

for(j=0;j<bmpWidth-wid;j++)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=\*(pBmpBuf+(i-hei)\*lineByte+(j+wid)\*3+k);

}

}

}

else

{

for(j=bmpWidth-1;j>=wid;j--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=\*(pBmpBuf+(i-hei)\*lineByte+(j-wid)\*3+k);

}

}

}

}

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

{

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

{

for(int k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=0;

}

}

}

if(ywid>0)

{

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

{

for(i=bmpHeight-1;i>=hei;i--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=0;

}

}

}

}

else

{

for(j=bmpWidth-1;j>=bmpWidth-wid;j--)

{

for(i=bmpHeight-1;i>=hei;i--)

{

for(k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

\*(pBmpBuf+i\*lineByte+j\*3+k)=0;

}

}

}

}

}

/\* for(int i=0;i<bmpHeight;i++)

{

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

{

for(int k=0;k<3;k++)//每像素RGB三个分量分别置0才变成黑色

{

//\*(pBmpBuf+i\*lineByte+j\*3+k)-=40;

m=\*(pBmpBuf+i\*lineByte+j\*3+k);

outfile<<m<<" ";

count\_xiang\_su++;

if(count\_xiang\_su%8==0)

{

outfile<<endl;

}

//n++;

}

n++;

}

}

cout<<"总的像素个素为:"<<n<<endl;

cout<<"----------------------------------------------------"<<endl; \*/

cout<<"平移图片完成！";

}

else

{

cout<<"输入错误,仍为原图\n";

}

//将图像数据存盘

char writePath[]="nvcpy.BMP";//图片处理后再存储

saveBmp(writePath, pBmpBuf, bmpWidth, bmpHeight, biBitCount, pColorTable);

//清除缓冲区，pBmpBuf和pColorTable是全局变量，在文件读入时申请的空间

delete []pBmpBuf;

if(biBitCount==8)

delete []pColorTable;

}

int main()

{

doIt();

cout<<"\n----------------------------------------------------"<<endl;

return 0;

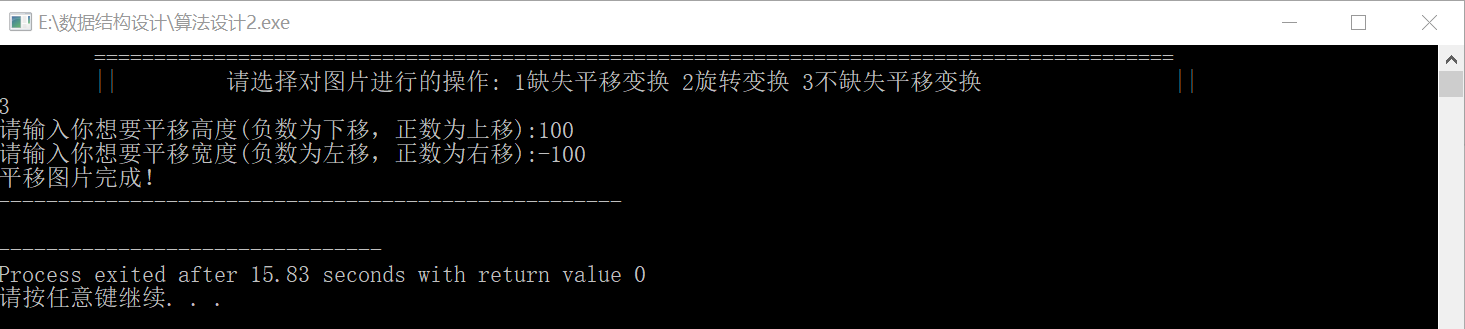
}

1. **测试数据及程序运行情况**

原图

****

**1、平移变换（不缺失平移）**

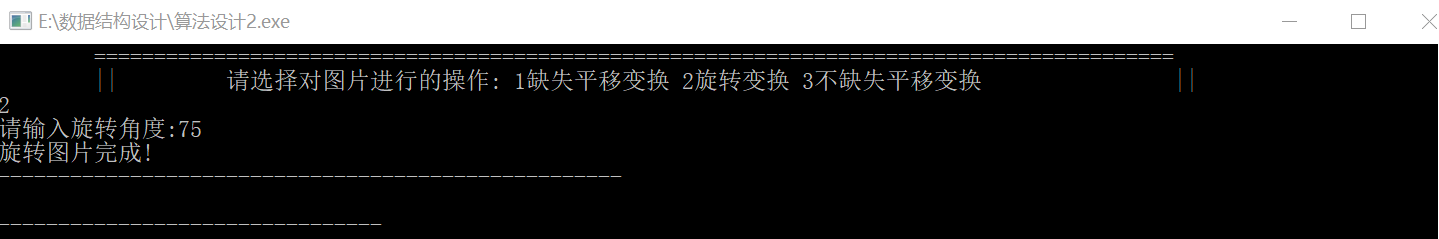


****

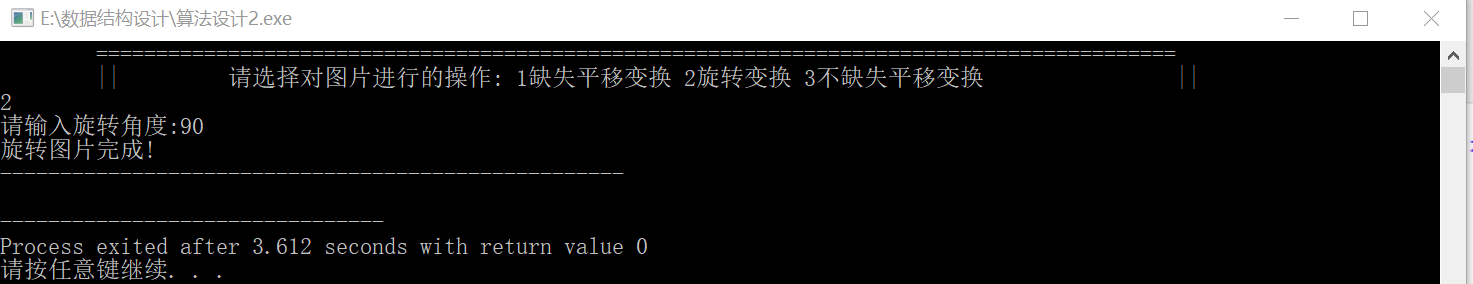
**2、平移变换（缺失平移）**

****

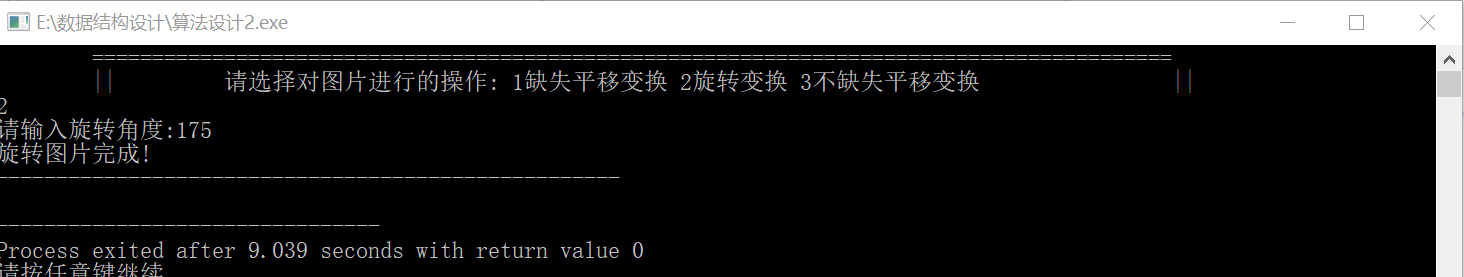
1. **旋转变换**



****



****



****

**五、实验过程中出现的问题及解决方法**

在实验中，我无法有效使用线性差值法使图片在旋转后仍保持与原版几乎无差别的清晰度，所以我采取了就近取值法，所以使得图片不像原来一样平缓。

**六、自我评析与总结**

通过此次课程设计，我复习了很多c++的知识，并对bmp图有了更加深刻的认识，对图像变换也有很多理解，尤其是旋转变换，让我经历了一次完整的思索求解过程，对图像变换认识也更加深刻。

**七、参考文献**

### 吕凤军 《数字图像处理编程入门》 清华大学出版社