**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

Факультет физико-математических и естественных наук

ОТЧЕТ

по лабораторной работе № 4

дисциплина: Компьютерная графика

Студент: Дорук Мерич Каймакджыоглу

Группа: НКНбд-01-20

**МОСКВА**

2022 г.

**Цел работу:** Модернизировать компьютерную программу для построения изображения выпуклого трехмерного тела в виде каркасной модели, разработанную при выполнении лабораторной работы № 3. Программа должна соответствовать следующим требованиям:

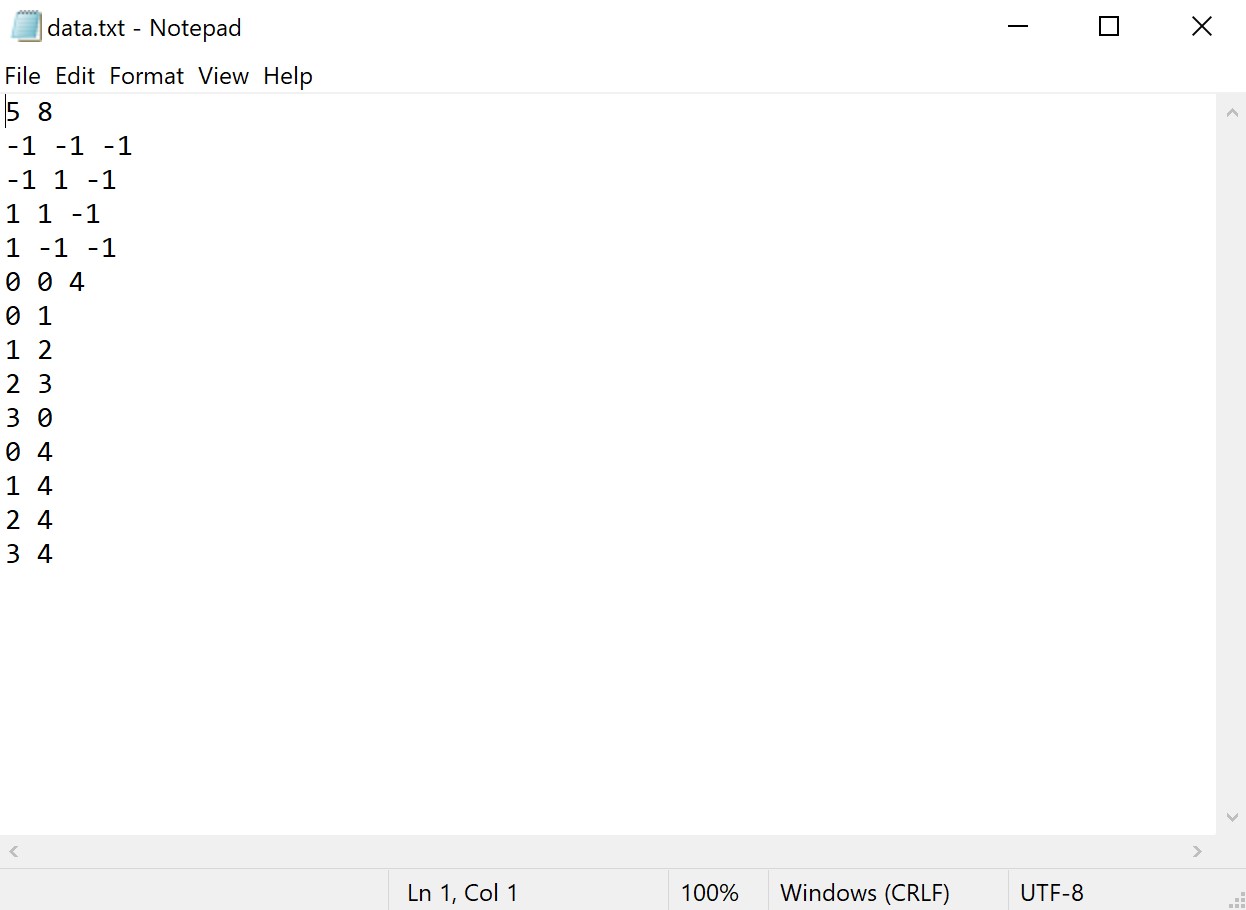
**Ход работы:**

- Загружать данные из файла описания объекта типа .dat, состоящего из трех разделов – описание координат вершин, описание ребер и описание треугольных граней. Файл может содержать данные о числе вершин, числе ребер и числе треугольных граней;

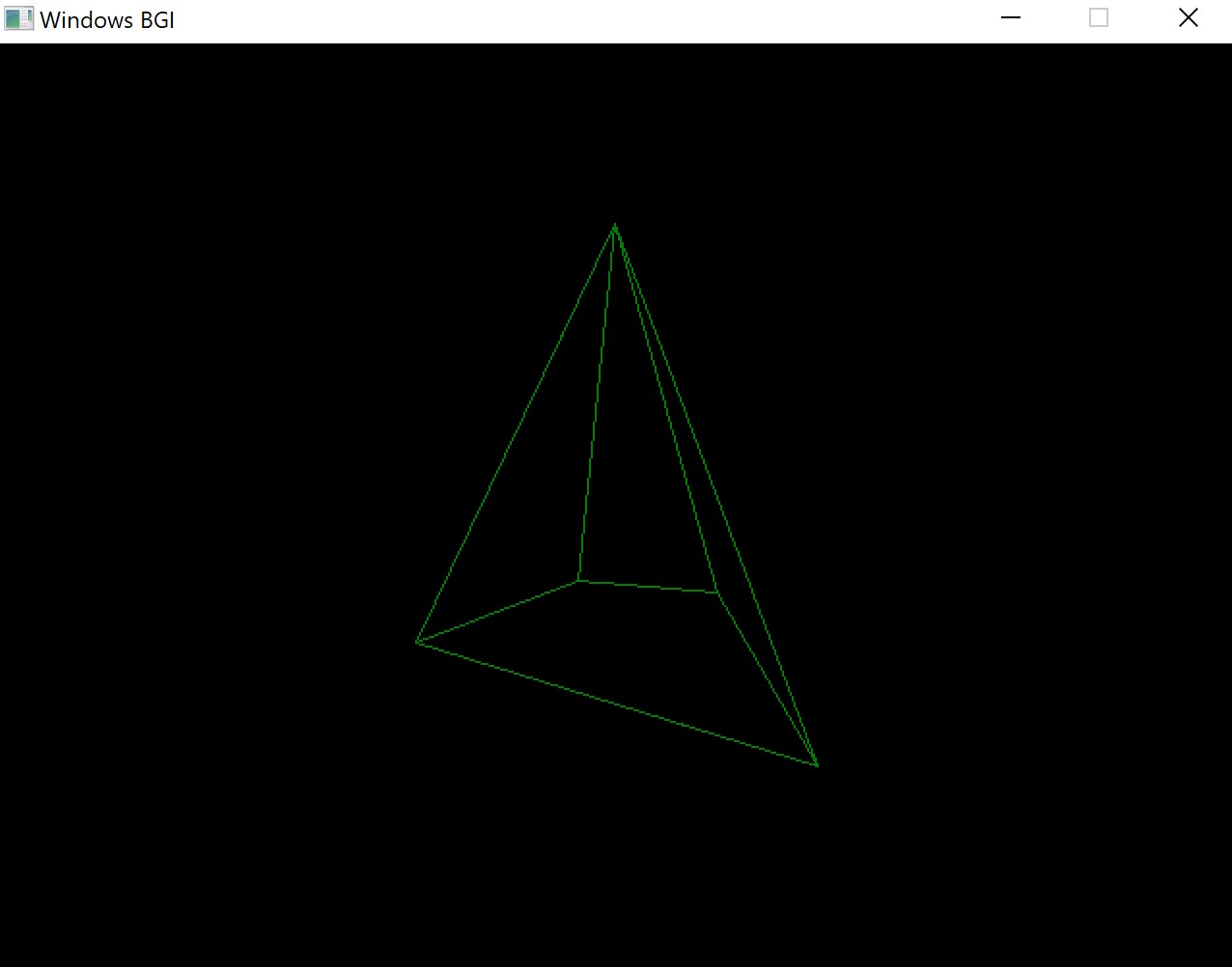
- Объектная модель должна быть дополнена описанием треугольных граней. Соответствующий класс должен содержать метод, выявляющий лицевые грани;

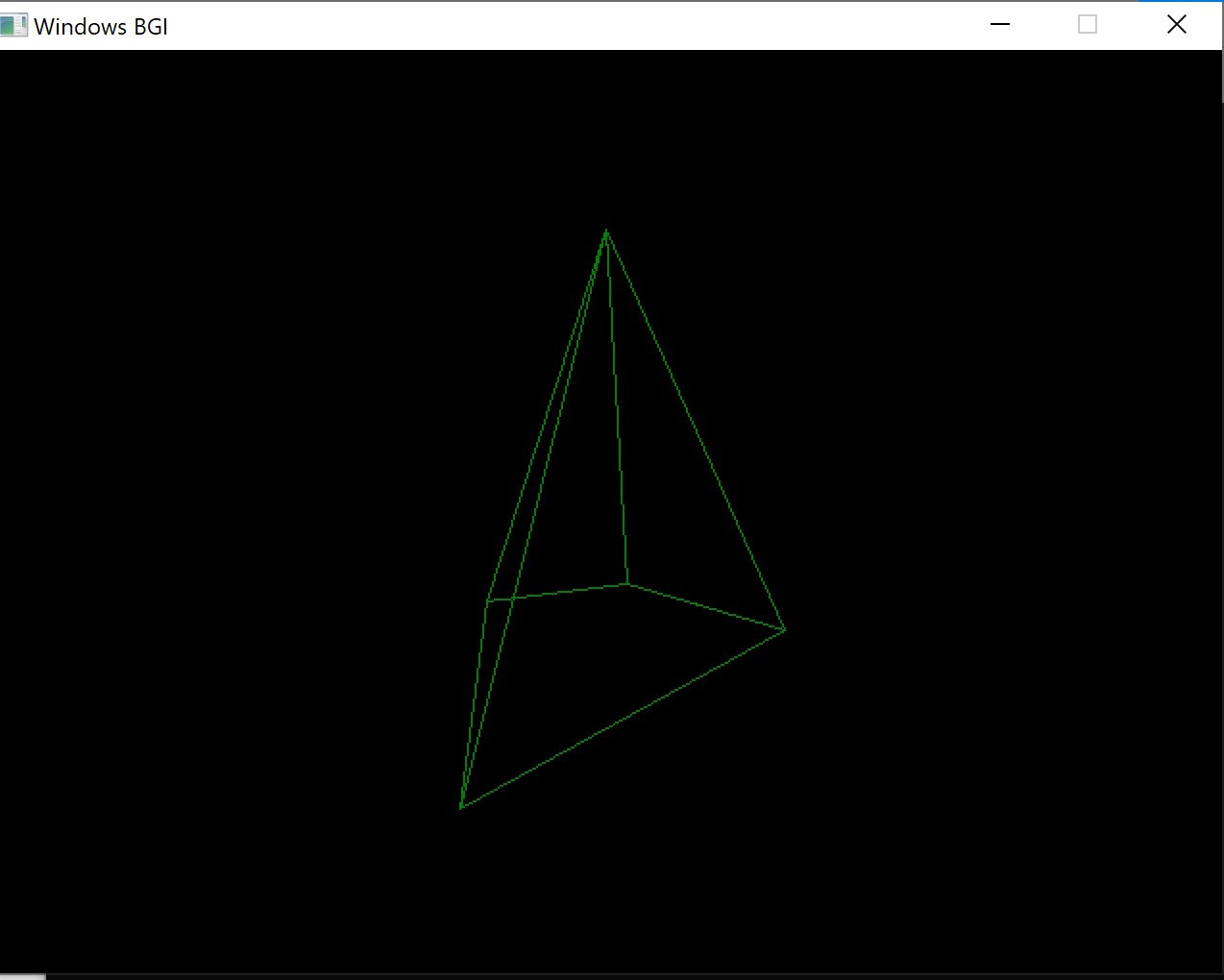
- Метод, отвечающий за построение изображения, должен быть дополнен алгоритмом удаления нелицевых граней;

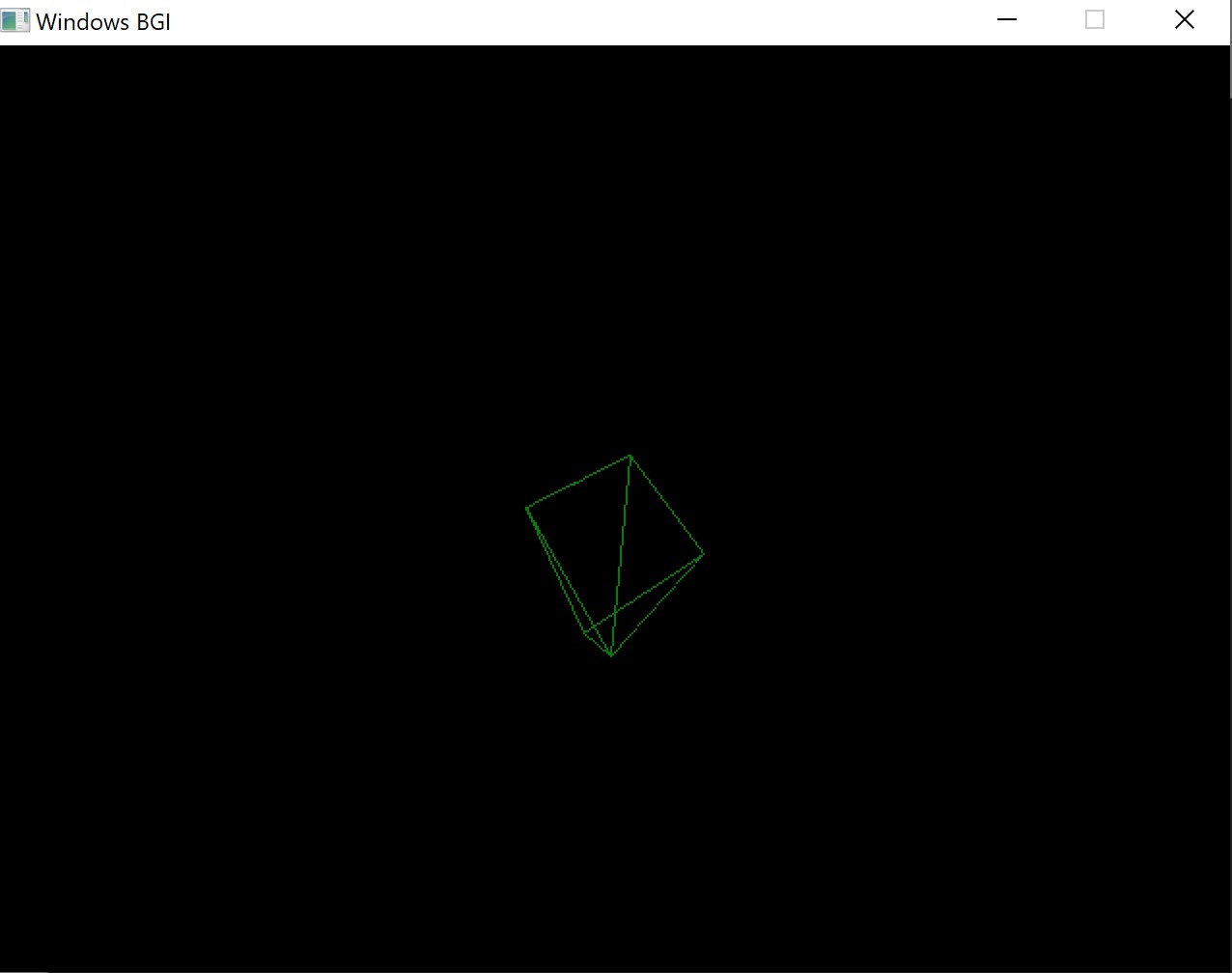
- Должны быть построены изображения куба (тестовое изображение) и выпуклого объекта, спроектированного автором программы.

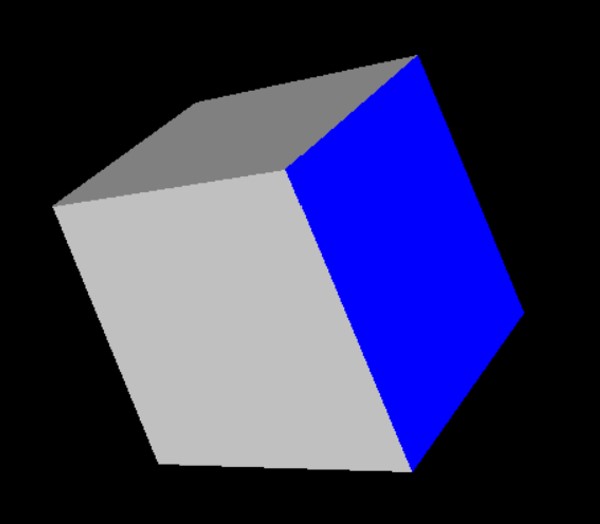


**Вывод:**

****

****

****

****

**КОД:**

#include "graphics.h"

#include <stdlib.h>

#include <time.h>

#include <Windows.h>

#include <math.h>

#include <iostream>

#include <fstream>

#include <windows.h>

using *namespace* std;

*class* tocka

{

*public:*

*double* x,y,z;

        tocka(){x=0;y=0;z=0;};

    tocka(*double* *a*,*double* *b*,*double* *c*){x=a;y=b;z=c;};

};

*class* vertex

{

*public:*

        tocka mir;

    tocka vid;

        vertex(){};

        vertex1(tocka *p*){mir=p;

        };

        ~vertex(){};

*void* SetViewCoord(tocka *sphercoord*){

*double* Ro=sphercoord.x,teta=sphercoord.y,fi1=sphercoord.z;

        vid.x=-mir.x\*sin(teta)+mir.y\*cos(teta);

        vid.y=-mir.x\*cos(fi1)\*cos(teta)-mir.y\*cos(fi1)\*sin(teta)+mir.z\*sin(fi1);

        vid.z=-mir.x\*sin(fi1)\*cos(teta)-mir.y\*sin(fi1)\*sin(teta)-mir.z\*cos(fi1)+Ro;

        };

};

*class* edge

{

*private:*

    vertex sv, fv;

*public:*

        edge(){};

        edge1 (vertex sver, vertex fver){

        sv=sver;fv=fver;

        }

        ~edge(){};

*void* DRAWedge(*double* *d*){

            d=d/2;

*double* x1 = (*int*)((d\*sv.vid.x)/sv.vid.z);

*double* y1 = (*int*)((d\*sv.vid.y)/sv.vid.z);

*double* x2 = (*int*)((d\*fv.vid.x)/fv.vid.z);

*double* y2 = (*int*)((d\*fv.vid.y)/fv.vid.z);

                line(x1+(getmaxx()/2),y1+(getmaxy()/2),x2+(getmaxx()/2),y2+(getmaxy()/2));

  }

};

*struct* eh

{

*public:*

*int* a,b;

};

*class* DRAW //edge vertex

{

*private:*

    vertex \*verts; edge \*edges; eh \*eh1;

    tocka vidpoint;

*bool* flg=0;

*public:*

*int* n,m;

    friend  *class* edge;

    friend  *class* vertex;

*void* draw(){

        for(*int* i=0;i<m;i++)

        edges[i].DRAWedge(vidpoint.x);

    };

*void* load(){

*int* c,a,b;

    if(!flg){

    FILE \*file;

    file = fopen("data.txt", "r");

  if (file!=NULL) {

    fscanf(file,"%d",&n);

    fscanf(file,"%d",&m);

        verts=new vertex[n];

        edges=new edge[m];

    for(*int* i=0;i<n;i++){

    fscanf(file,"%d",&a);

    fscanf(file,"%d",&b);

    fscanf(file,"%d",&c);

    tocka p=tocka(a\*100,b\*100,c\*100);

    verts[i].vertex1(p);

    verts[i].SetViewCoord(vidpoint);

    }

        eh1=new eh[m];

        for(*int* i=0;i<m;i++){

    fscanf(file,"%d",&a);

    fscanf(file,"%d",&b);

    eh1[i].a=a;

    eh1[i].b=b;

    edges[i].edge1(verts[eh1[i].a],verts[eh1[i].b]);

    }

    flg=1;

    }}

for(*int* i=0;i<n;i++)verts[i].SetViewCoord(vidpoint);

    for(*int* i=0;i<m;i++)

    edges[i].edge1(verts[eh1[i].a],verts[eh1[i].b]);

    };

*void* Setvidimtck(*double* *rota*,*double* *alp*,*double* *fi1*){

    vidpoint.x=rota;

    vidpoint.y=alp;

    vidpoint.z=fi1;

    };

*void* drawsurface(){

    };

*void* delObject(){};

};

*void* um(*double*\*\* *tc*, *double*\*\* *d*)  //tc

{

*double*\*\* ta;

ta=new *double*\*[8];

for (*int* i=0;i<8;i++)

ta[i]=new *double*[4];

for (*int* i=0;i<8;i++)

for (*int* j=0;j<4;j++)

ta[i][j]=0;

  for (*int* i=0;i<8;i++)

    for (*int* j=0;j<4;j++)

      for (*int* k=0;k<4;k++)

         ta[i][j]=ta[i][j]+tc[i][k]\*d[k][j];

    for (*int* i=0;i<8;i++)

    {//cout<<endl;

    for (*int* j=0;j<4;j++)

    {tc[i][j]=ta[i][j];

    //cout<<tc[i][j]<<" ";

    }

    }

cout<<endl;

}

*void* umm(*double*\*\* *tc*, *double*\*\* *d*, *double*\*\* *tb*) //tb

{

*double*\*\* ta;

ta=new *double*\*[8];

for (*int* i=0;i<8;i++)

ta[i]=new *double*[4];

for (*int* i=0;i<8;i++)

for (*int* j=0;j<4;j++)

ta[i][j]=0;

  for (*int* i=0;i<8;i++)

    for (*int* j=0;j<4;j++)

      for (*int* k=0;k<4;k++)

         ta[i][j]=ta[i][j]+tc[i][k]\*d[k][j];

    for (*int* i=0;i<8;i++)

    {//cout<<endl;

    for (*int* j=0;j<4;j++)

    {tb[i][j]=ta[i][j];

    //cout<<tc[i][j]<<" ";

    }

    }

cout<<endl;

}

*double* bv(*int* *t1*, *int* *t2*, *int* *t3*, *double*\*\* *tc*) //t1t2t3

{

*double* h,h1,h2,h3;

h1=tc[t1][0]\*(tc[t2][1]\*tc[t3][2]-tc[t3][1]\*tc[t2][2]);

h2=-tc[t2][0]\*(tc[t1][1]\*tc[t3][2]-tc[t3][1]\*tc[t1][2]);

h3=tc[t3][0]\*(tc[t1][1]\*tc[t2][2]-tc[t2][1]\*tc[t1][2]);

h=h1+h2+h3;

return h;

}

*double* ugl(*int* *t1*, *int* *t2*, *int* *t3*, *double*\*\* *tc*) // t1t2t3.

{

*double* h,a,b,c,al,znk,p,xc,yc,zc,t;

h=bv(t1,t2,t3,tc);

a=tc[t1][1]\*(tc[t2][2]-tc[t3][2])-tc[t2][1]\*(tc[t1][2]-tc[t3][2])+tc[t3][1]\*(tc[t1][2]-tc[t2][2]);      //a

b=-(tc[t1][0]\*(tc[t2][2]-tc[t3][2])-tc[t2][0]\*(tc[t1][2]-tc[t3][2])+tc[t3][0]\*(tc[t1][2]-tc[t2][2]));   // b

c=tc[t1][0]\*(tc[t2][1]-tc[t3][1])-tc[t2][0]\*(tc[t1][1]-tc[t3][1])+tc[t3][0]\*(tc[t1][1]-tc[t2][1]);      //c

xc=(tc[t1][0]+tc[t2][0]+tc[t3][0])/3; //x

yc=(tc[t1][1]+tc[t2][1]+tc[t3][1])/3; //y

zc=(tc[t1][2]+tc[t2][2]+tc[t3][2])/3; //z

p=a\*xc+b\*yc+c\*zc-h;

if (p>=0) znk=1;

else znk=-1;

t=sqrt(a\*a+b\*b+c\*c);

a=a/t;

b=b/t;

c=c/t;

al=znk\*((xc\*a+yc\*b+zc\*c)/(sqrt(xc\*xc+yc\*yc+zc\*zc)\*sqrt(a\*a+b\*b+c\*c)));

return al;

}

*void* paint(*double*\*\* *tc*, *double* *r*, *double*\*\* *h*, *double*\*\* *all*)

{

setbkcolor(0);

cleardevice();

*double*\* x;

*double*\* y;

x=new *double*[8];

y=new *double*[8];

setcolor(15);

for (*int* i=0;i<8;i++)

{

cout<<endl;

x[i]=r\*tc[i][0]/(2\*tc[i][2]);

y[i]=r\*tc[i][1]/(2\*tc[i][2]);

}

line(x[0],y[0],x[1],y[1]);

line(x[0],y[0],x[2],y[2]);

line(x[0],y[0],x[4],y[4]);

line(x[1],y[1],x[3],y[3]);

line(x[1],y[1],x[5],y[5]);

line(x[2],y[2],x[6],y[6]);

line(x[2],y[2],x[3],y[3]);

line(x[3],y[3],x[7],y[7]);

line(x[4],y[4],x[5],y[5]);

line(x[4],y[4],x[6],y[6]);

line(x[5],y[5],x[7],y[7]);

line(x[6],y[6],x[7],y[7]);

*int* shape0[] =

{x[0], y[0],

x[1], y[1],

x[5], y[5],

x[4], y[4]};

*int* shape1[] =

{x[1], y[1],

x[3], y[3],

x[7], y[7],

x[5], y[5]};

*int* shape2[] =

{x[4], y[4],

x[5], y[5],

x[7], y[7],

x[6], y[6]};

*int* shape3[] =

{x[2], y[2],

x[3], y[3],

x[7], y[7],

x[6], y[6]};

*int* shape4[] =

{x[0], y[0],

x[2], y[2],

x[6], y[6],

x[4], y[4]};

*int* shape5[] =

{x[0], y[0],

x[1], y[1],

x[3], y[3],

x[2], y[2]};

if (h[0][0]>0 and h[0][1]>0)

{   setcolor(4);

    setfillstyle(SOLID\_FILL, 4);

    fillpoly(4, shape0);}

if (h[1][0]>0 and h[1][1]>0)

{   setcolor(3);

    setfillstyle(SOLID\_FILL, 3);

    fillpoly(4, shape1);}

if (h[2][0]>0 and h[2][1]>0)

{   setcolor(2);

    setfillstyle(SOLID\_FILL, 2);

    fillpoly(4, shape2);}

if (h[3][0]>0 and h[3][1]>0)

{   setcolor(8);

    setfillstyle(SOLID\_FILL, 8);

    fillpoly(4, shape3);}

if (h[4][0]>0 and h[4][1]>0)

{   setcolor(1);

    setfillstyle(SOLID\_FILL, 1);

    fillpoly(4, shape4);}

if (h[5][0]>0 and h[5][1]>0)

{   setcolor(14);

    setfillstyle(SOLID\_FILL, 14);

    fillpoly(4, shape5);}

}

*int* main()

{

*double* rota=200,alp=0,fi1=0;

*int* tmp;

*int* gddriver = DETECT, gmode, errorcode;

    initgraph(&gddriver, &gmode, "");

    setfillstyle(LINE\_FILL,BLACK) ;

    DRAW cub;

    cub.Setvidimtck(rota,alp,fi1);

    cub.load();

    cub.draw();

while(true)   {

        cleardevice();

        setcolor(2);

        if(tmp=='a')

        alp += ((M\_PI/9));

        if(tmp=='d')

        alp -= ((M\_PI/9));

        if(tmp=='w')

        fi1 += ((M\_PI/9));

        if(tmp=='s')

        fi1 -= ((M\_PI/9));

        if(tmp=='e')

        rota +=6;

        if(tmp=='q')

        rota -=6;

        if(tmp==27)

        break;

        cub.Setvidimtck(rota,alp,fi1);

        cub.load();

        cub.draw();

        tmp = getch();

    }

*double* k;

*double*\*\* tb;

tb=new *double*\*[8];

for (*int* i=0;i<8;i++)

tb[i]=new *double*[4];

*double*\*\* tc;

tc=new *double*\*[8];

for (*int* i=0;i<8;i++)

tc[i]=new *double*[4];

tc[0][0]=1;     tc[0][1]=-1;    tc[0][2]=-1;    tc[0][3]=1;

tc[1][0]=1;     tc[1][1]=1;     tc[1][2]=-1;    tc[1][3]=1;

tc[2][0]=-1;    tc[2][1]=-1;    tc[2][2]=-1;    tc[2][3]=1;

tc[3][0]=-1;    tc[3][1]=1;     tc[3][2]=-1;    tc[3][3]=1;

tc[4][0]=1;     tc[4][1]=-1;    tc[4][2]=1;     tc[4][3]=1;

tc[5][0]=1;     tc[5][1]=1;     tc[5][2]=1;     tc[5][3]=1;

tc[6][0]=-1;    tc[6][1]=-1;    tc[6][2]=1;     tc[6][3]=1;

tc[7][0]=-1;    tc[7][1]=1;     tc[7][2]=1;     tc[7][3]=1;

k=200;

for (*int* i=0;i<8;i++)

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

{tc[i][j]=tc[i][j]\*k;}

*double* r=2000, tau=0, fi2=0;

*double*\*\* b;

b=new *double*\*[4];

for (*int* i=0;i<4;i++)

b[i]=new *double*[4];

b[0][0]=-sin(tau);  b[0][1]=-cos(fi2)\*cos(tau); b[0][2]=-sin(fi2)\*cos(tau); b[0][3]=0;

b[1][0]=cos(tau);   b[1][1]=-cos(fi2)\*sin(tau); b[1][2]=-sin(fi2)\*sin(tau); b[1][3]=0;

b[2][0]=0;          b[2][1]=sin(fi2);           b[2][2]=-cos(fi2);          b[2][3]=0;

b[3][0]=0;          b[3][1]=0;                  b[3][2]=r;                  b[3][3]=1;

*double* dx=-500,dy=500,dz=100;

*double*\*\* p;

p=new *double*\*[4];

for (*int* i=0;i<4;i++)

p[i]=new *double*[4];

p[0][0]=1;  p[0][1]=0;  p[0][2]=0;  p[0][3]=0;

p[1][0]=0;  p[1][1]=1;  p[1][2]=0;  p[1][3]=0;

p[2][0]=0;  p[2][1]=0;  p[2][2]=1;  p[2][3]=0;

p[3][0]=dx; p[3][1]=dy; p[3][2]=dz; p[3][3]=1;

*double*\*\* xp;

xp=new *double*\*[4];

for (*int* i=0;i<4;i++)

xp[i]=new *double*[4];

*double* al=3.14159/4;

xp[0][0]=1; xp[0][1]=0;         xp[0][2]=0;         xp[0][3]=0;

xp[1][0]=0; xp[1][1]=cos(al);   xp[1][2]=sin(al);   xp[1][3]=0;

xp[2][0]=0; xp[2][1]=-sin(al);  xp[2][2]=cos(al);   xp[2][3]=0;

xp[3][0]=0; xp[3][1]=0;         xp[3][2]=0;         xp[3][3]=1;

*double*\*\* yp;

yp=new *double*\*[4];

for (*int* i=0;i<4;i++)

yp[i]=new *double*[4];

al=3.14159/4;

yp[0][0]=cos(al);   yp[0][1]=0; yp[0][2]=sin(al);   yp[0][3]=0;

yp[1][0]=0;         yp[1][1]=1; yp[1][2]=0;         yp[1][3]=0;

yp[2][0]=-sin(al);  yp[2][1]=0; yp[2][2]=cos(al);   yp[2][3]=0;

yp[3][0]=0;         yp[3][1]=0; yp[3][2]=0;         yp[3][3]=1;

*double*\*\* zp;

zp=new *double*\*[4];

for (*int* i=0;i<4;i++)

zp[i]=new *double*[4];

al=3.14159/8;

zp[0][0]=cos(al);   zp[0][1]=sin(al);   zp[0][2]=0;     zp[0][3]=0;

zp[1][0]=-sin(al);  zp[1][1]=cos(al);   zp[1][2]=0;     zp[1][3]=0;

zp[2][0]=0;         zp[2][1]=0;         zp[2][2]=1;     zp[2][3]=0;

zp[3][0]=0;         zp[3][1]=0;         zp[3][2]=0;     zp[3][3]=1;

*double*\*\* h;

h=new *double*\*[6];

for (*int* i=0;i<6;i++)

h[i]=new *double*[2];

h[0][0]=bv(0,1,5,tc);

h[0][1]=bv(0,5,4,tc);

h[1][0]=bv(1,3,7,tc);

h[1][1]=bv(1,7,5,tc);

h[2][0]=bv(4,5,7,tc);

h[2][1]=bv(4,7,6,tc);

h[3][0]=bv(2,7,3,tc);

h[3][1]=bv(2,6,7,tc);

h[4][0]=bv(0,6,2,tc);

h[4][1]=bv(0,4,6,tc);

h[5][0]=bv(0,3,1,tc);

h[5][1]=bv(0,2,3,tc);

//cout<<h1;

//um(tc,xp);

//um(tc,zp);

*double*\*\* all;

all=new *double*\*[6];

for (*int* i=0;i<6;i++)

all[i]=new *double*[2];

all[0][0]=ugl(0,1,5,tc);

all[0][1]=ugl(0,5,4,tc);

all[1][0]=ugl(1,3,7,tc);

all[1][1]=ugl(1,7,5,tc);

all[2][0]=ugl(4,5,7,tc);

all[2][1]=ugl(4,7,6,tc);

all[3][0]=ugl(2,7,3,tc);

all[3][1]=ugl(2,6,7,tc);

all[4][0]=ugl(0,6,2,tc);

all[4][1]=ugl(0,4,6,tc);

all[5][0]=ugl(0,3,1,tc);

all[5][1]=ugl(0,2,3,tc);

umm(tc,p,tb);

umm(tb,b,tb);

paint(tb,r,h,all);

*int* i=0;

while(i==0)

{*char* c;

c=getch();

if (c==50)

    {

    al=3.14159/4;

    um(tc,yp);

    umm(tc,p,tb);

    umm(tb,b,tb);

    h[0][0]=bv(0,1,5,tb);

    h[0][1]=bv(0,5,4,tb);

    h[1][0]=bv(1,3,7,tb);

    h[1][1]=bv(1,7,5,tb);

    h[2][0]=bv(4,5,7,tb);

    h[2][1]=bv(4,7,6,tb);

    h[3][0]=bv(2,7,3,tb);

    h[3][1]=bv(2,6,7,tb);

    h[4][0]=bv(0,6,2,tb);

    h[4][1]=bv(0,4,6,tb);

    h[5][0]=bv(0,3,1,tb);

    h[5][1]=bv(0,2,3,tb);

all[0][0]=ugl(0,1,5,tb);

all[0][1]=ugl(0,5,4,tb);

all[1][0]=ugl(1,3,7,tb);

all[1][1]=ugl(1,7,5,tb);

all[2][0]=ugl(4,5,7,tb);

all[2][1]=ugl(4,7,6,tb);

all[3][0]=ugl(2,7,3,tb);

all[3][1]=ugl(2,6,7,tb);

all[4][0]=ugl(0,6,2,tb);

all[4][1]=ugl(0,4,6,tb);

all[5][0]=ugl(0,3,1,tb);

all[5][1]=ugl(0,2,3,tb);

    paint(tb,r,h,all);}

if (c==54)

    {

    //cout<<"left";

    al=3.14159/4;

    um(tc,xp);

    umm(tc,p,tb);

    umm(tb,b,tb);

    h[0][0]=bv(0,1,5,tb);

    h[0][1]=bv(0,5,4,tb);

    h[1][0]=bv(1,3,7,tb);

    h[1][1]=bv(1,7,5,tb);

    h[2][0]=bv(4,5,7,tb);

    h[2][1]=bv(4,7,6,tb);

    h[3][0]=bv(2,7,3,tb);

    h[3][1]=bv(2,6,7,tb);

    h[4][0]=bv(0,6,2,tb);

    h[4][1]=bv(0,4,6,tb);

    h[5][0]=bv(0,3,1,tb);

    h[5][1]=bv(0,2,3,tb);

all[0][0]=ugl(0,1,5,tb);

all[0][1]=ugl(0,5,4,tb);

all[1][0]=ugl(1,3,7,tb);

all[1][1]=ugl(1,7,5,tb);

all[2][0]=ugl(4,5,7,tb);

all[2][1]=ugl(4,7,6,tb);

all[3][0]=ugl(2,7,3,tb);

all[3][1]=ugl(2,6,7,tb);

all[4][0]=ugl(0,6,2,tb);

all[4][1]=ugl(0,4,6,tb);

all[5][0]=ugl(0,3,1,tb);

all[5][1]=ugl(0,2,3,tb);

    paint(tb,r,h,all);}

if (c==56)

    {

    al=3.14159/4;

    um(tc,zp);

    umm(tc,p,tb);

    umm(tb,b,tb);

    h[0][0]=bv(0,1,5,tb);

    h[0][1]=bv(0,5,4,tb);

    h[1][0]=bv(1,3,7,tb);

    h[1][1]=bv(1,7,5,tb);

    h[2][0]=bv(4,5,7,tb);

    h[2][1]=bv(4,7,6,tb);

    h[3][0]=bv(2,7,3,tb);

    h[3][1]=bv(2,6,7,tb);

    h[4][0]=bv(0,6,2,tb);

    h[4][1]=bv(0,4,6,tb);

    h[5][0]=bv(0,3,1,tb);

    h[5][1]=bv(0,2,3,tb);

all[0][0]=ugl(0,1,5,tb);

all[0][1]=ugl(0,5,4,tb);

all[1][0]=ugl(1,3,7,tb);

all[1][1]=ugl(1,7,5,tb);

all[2][0]=ugl(4,5,7,tb);

all[2][1]=ugl(4,7,6,tb);

all[3][0]=ugl(2,7,3,tb);

all[3][1]=ugl(2,6,7,tb);

all[4][0]=ugl(0,6,2,tb);

all[4][1]=ugl(0,4,6,tb);

all[5][0]=ugl(0,3,1,tb);

all[5][1]=ugl(0,2,3,tb);

paint(tb,r,h,all);

    }

if (c==13) break;

}

    getch();

    closegraph () ;

    return 0;

}