# ЛАБОРАТОРНАЯ РАБОТА № 1
# ГЕОМЕТРИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ НА ПЛОСКОСТИ

**Цель работы:** составить программу для преобразований двухмерного геометрического объекта на плоскости. Продемонстрировать результаты на экране компьютера.

## Краткие теоретические сведения

В компьютерной графике широко используются геометрические преобразования, как двухмерные, так и трехмерные. Необходимость в преобразованиях возникает в различных ситуациях: при переходе от декартовых координат изображаемого объекта к физическим координатам устройства; при получении различиях видов, разрезов или сечений детали; при построении наглядных изображений объекта в перспективе или аксонометрии; при перекосе, масштабировании и повороте изображений. Все эти преобразования можно выполнить на основе математических методов.

При выполнении лабораторной работы решаются задачи формирования исходного изображения плоского объекта с использованием графического расширения алгоритмического языка высокого уровня (Turbo Pascal, C, C++, Delphi); программной реализации математических методов геометрических преобразований на плоскости.

## Математические методы

### Двухмерные преобразования

Перенос – смещение графических примитивов на один и тот же вектор. При перенесении изображения без поворота связь между перемещенной точкой и ее первоначальным положением выражается уравнениями (рис. 1.1):

$$x' = x + Dx, \quad y' = y + Dy,$$

или в векторной форме:

$$[x'\,y'] = [xy] + [DxDy] \quad \text{или} \quad P' = P + T.$$

Масштабирование. Точки изображения можно промасштабировать (сжать или растянуть) в $S_x$ раз вдоль оси $X$ и в $S_y$ раз вдоль оси $Y$, в результате получаются новые точки:

$$x' = x \cdot S_{x,} \quad y' = y \cdot S_y.$$

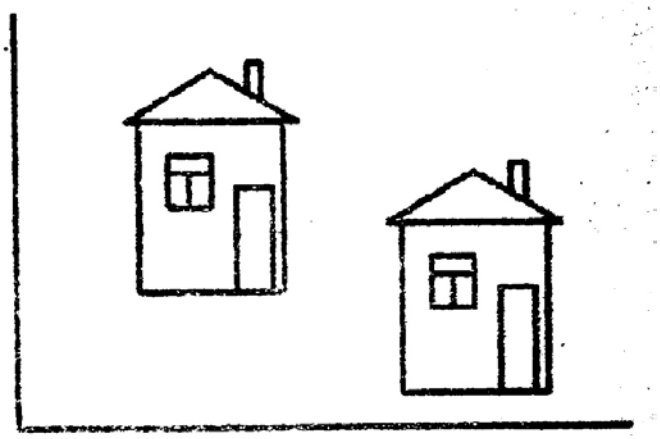


Рис. 1.1. Перенос объекта

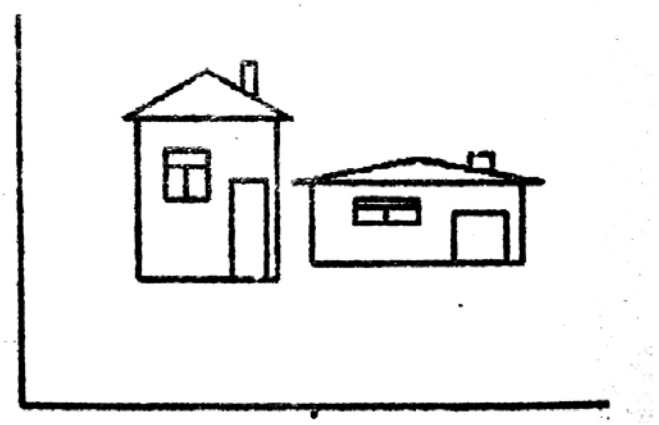


Рис. 1.2. Масштабирование объекта. $S_x = 1{,}5$, $S_y = 0{,}5$

Если определить $S$ как $\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$, то можно записать в матричной форме:

$$[x'y'] = [xy] \cdot \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

или $P' = P \cdot S$.

На рис. 1.2 показан домик, промасштабированный с коэффициентами 1,5 по оси $x$ и 0,5 по оси $y$. Масштабирование производилось относительно центральной точки домика. Пропорции домика изменились и на рис. 1.3. В обоих случаях было применено неоднородное масштабирование, при котором $S_x \neq S_y$. Однородное масштабирование, для которого $S_x = S_y$, не влияет на пропорции.

Поворот. Точки могут быть повернуты на угол θ относительно начала координат, как показано на рис. 1.4. Математически поворот определяется следующим образом:

$$x' = x \cdot \cos\theta - y \sin\theta,$$
$$y' = x \cdot \sin\theta + y \cos\theta,$$

в матричной форме:

$$[x'\,y'] = [xy] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

На рис. 1.5 показан пример поворота изображения относительно центральной его точки на угол θ = 45°.

## Однородные координаты

Описанные ранее преобразования переноса, масштабирования и поворота могут быть записаны в матричной форме:

$$P' = P + T,$$
$$P' = P \cdot S,$$
$$P' = P \cdot R.$$

Часто приходится сочетать эти элементарные виды преобразования по два и более раз. Между тем перенос реализуется с помощью сложения, а масштабирование и поворот – с помощью умножения. Для объединения этих преобразований целесообразно воспользоваться однородными координатами. С помощью таких координат все три преобразования можно реализовать при помощи умножения. Матричное произведение иногда называют композицией.

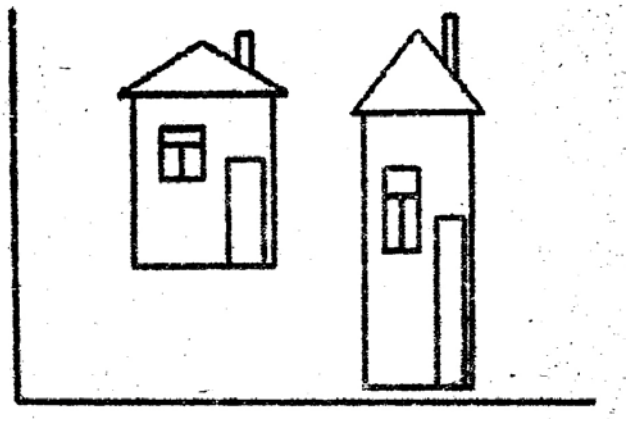


Рис. 1.3. Масштабирование объекта. $S_x$ = 0,75, $S_y$ = 1,6

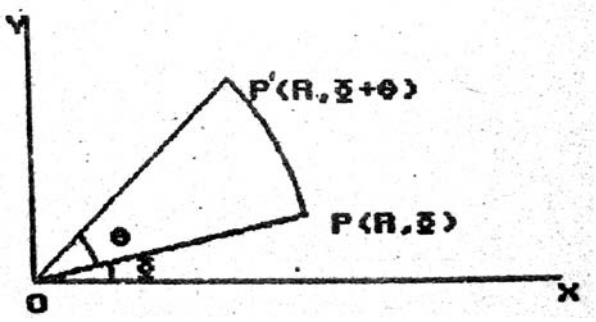



Рис. 1.4. Поворот точки

Точки в однородных координатах описываются трехэлементными вектор-строками. Чтобы получить другой вектор точки, матрица, на которую умножается вектор точки, должна иметь размер 3×3. Уравнения переноса з аписываются в виде матрицы преобразования однородных координат:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}$$

или

$$P' = P \cdot T(D_x, D_y),$$

где

$$T(D_x, D_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}.$$

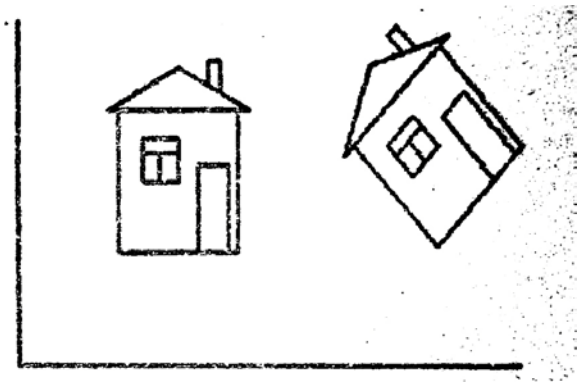


Рис. 1.5. Поворот объекта

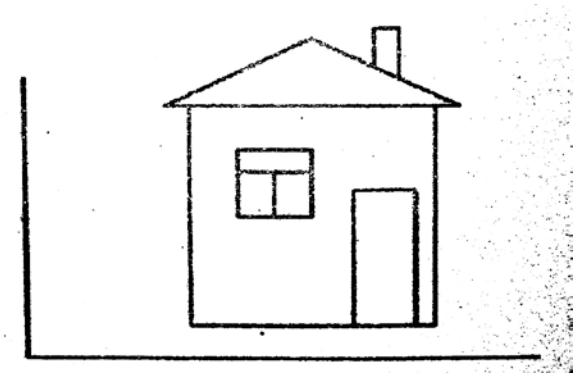


Рис. 1.6. Параметризация объекта

Запишем уравнения преобразования масштабирования в матричной форме:

$$[x'\,y'1]=[x \quad y \quad 1]\cdot\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Примем

$$S(S_x,S_y)=\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

и получим

$$P'=P\cdot S(S_x,S_y).$$

Уравнения преобразования поворота записываем в виде

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\cdot\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Принимая

$$R(\theta)=\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

окончательно получаем $P'=P\cdot R(\theta)$.

При повороте изображения вокруг произвольной точки $M$ на угол $\theta$ разбиваем задачу на три элементарных преобразования:

1. Перенос точки $M$ в начало координат.

2. Поворот вокруг начала координат.

3. Обратный перенос точки $M$ в первоначальное положение. Эта последовательность показана на рис. 1.5.

Первый перенос выполняется на $(-x_1, -y_1)$, а последующий – на $(x_1, y_1)$ – обратный ему.

Преобразование композиции выглядит так:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x_1(1-\cos\theta) + y_1\sin\theta & y_1(1-\cos\theta) - x_1\sin\theta & 1 \end{bmatrix}.$$

Из примера видно, как применение однородных координат упрощает решение задачи на композицию двухмерных преобразований.

## Порядок выполнения работы

Выполнение лабораторной работы можно разбить на три этапа:

1. Построение изображения с использованием базисных процедур и функций какого-либо алгоритмического языка.

2. Программная реализация какого-либо преобразования или цепочки (композиции) преобразований.

3. Оформление результатов работы.

Построение изображения. Исходный двухмерный объект может быть взят согласно варианту. Далее необходимо определить все опорные точки, задающие объект, как это показано на рис. 1.6. Массив координат опорных точек и связи между ними есть геометрическая модель объекта.

Ниже приводится пример программы для построения домика, написанной на языке TURBO PASCAL. Комментарии в программе позволяют легко понять назначение операторов и процедур языка.

```
program home;
        uses Graph;   {ИСПОЛЬЗУЕМЫЕ  МОДУЛИ}
var
        Gd, Gm: Integer;   {ОПИСАНИЕ ПЕРЕМЕННЫХ}
         x,y:array[1..23] of integer; {ОПИСАНИЕ МАССИВА ТОЧЕК}
begin
        Gd:=Detect; InitGraph (Gd, Gm,' '); {ИНИЦИАЛИЗАЦИЯ}
        if GraphResult () grOk then halt(1);  { ГРАФИЧЕСКОГО РЕЖИМА}
        <**************КРЫША************>
        Line(x[10],y[10],x[5],y[5]);   {ПРОЦЕДУРЫ  ВЫЧЕРЧИВАНИЯ}
```

```
        Line(x[5],y[5],x[11],y[11]);
        Line(x[10],y[10],x[11],y[11]);
        <***************ДОМ***************>
        Line(x[1],y[1],x[2],y[2]);
        Line(x[2],y[2],x[3],y[3]);
        Line(x[3],y[3],x[4],y[4]);
        Line(x[4],y[4],x[1],y[1]);
        <***************ТРУБА**************>
        Line(x[8],y[8],x[6],y[6]);
        Line(x[6],y[6],x[7],y[7]);
        Line(x[7],y[7],x[9],y[9]);
        <***************ОКНО**************>
        Line(x[12],y[12],x[14],y[14]);
        Line(x[14],y[14],x[15],y[15]);
        Line(x[15],y[15],x[17],y[17]);
        Line(x[17],y[17],x[12],y[12]);
        Line(x[18],y[18],x[19],y[19]);
        Line(x[13],y[13],x[16],y[16]);
        <***************ДВЕРЬ***************>
        Line(x[20],y[20],x[21],y[21]);
        Line(x[21],y[21],x[22],y[22]);
        Line(x[22],y[22],x[23],y[23]);
        CloseGraph;
End.
```

Преобразование изображения. Программу построения объекта необходимо теперь дополнить блоком, задающим геометрические преобразования. Для любого плоского объекта преобразование в однородных координатах в матричной форме можно записать в следующем виде:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ . & . & . \\ x_n & y_n & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} x_1' & y_1' & 1 \\ x_2' & y_2' & 1 \\ x_3' & y_3' & 1 \\ . & . & . \\ x_n' & y_n' & 1 \end{bmatrix}$$

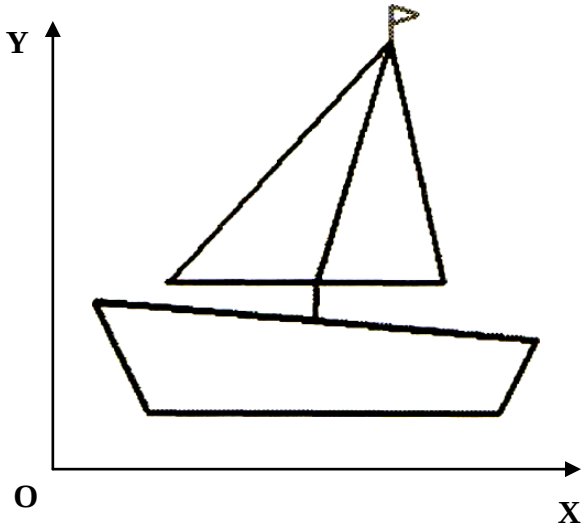
В этой записи первая матрица представляет собой массив координат точек, описывающих исходный объект. Матрица преобразований имеет размер 3×3. Выбирая необходимые значения элементов матрицы, можно реализовать то или иное геометрическое преобразование. Более того, цепочка

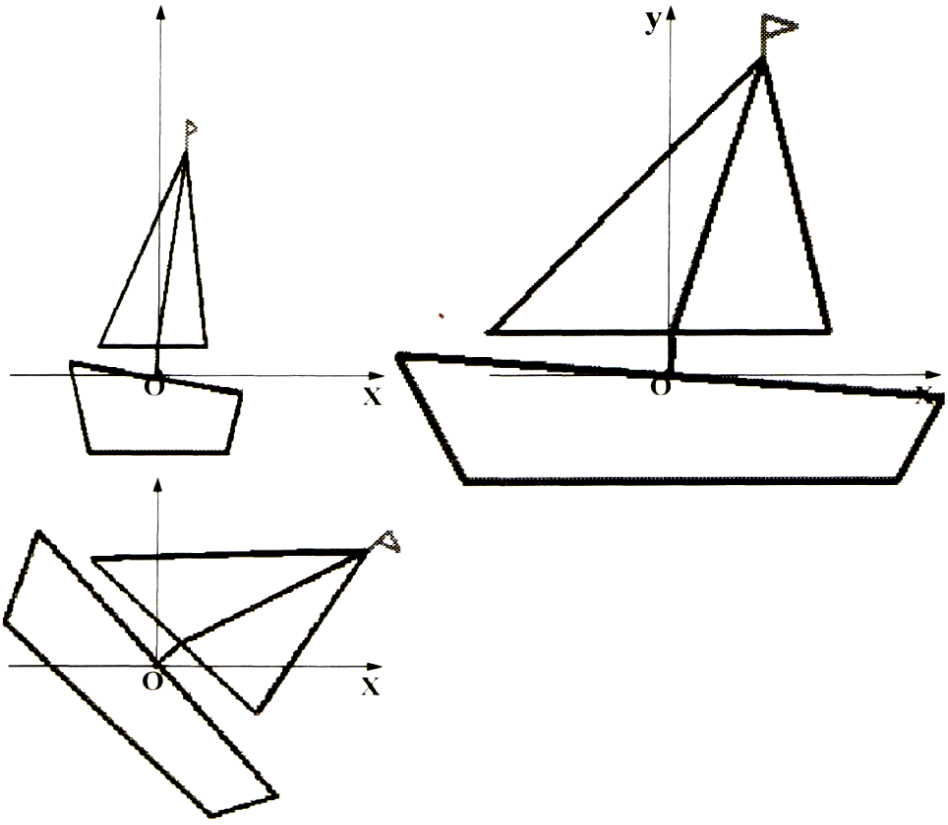
(композиция) преобразований выполняется такой же операцией – умножением матрицы с исходными координатами точек на результирующую матрицу преобразований размером 3×3. В результате получается третья матрица с новыми координатами точек объекта.

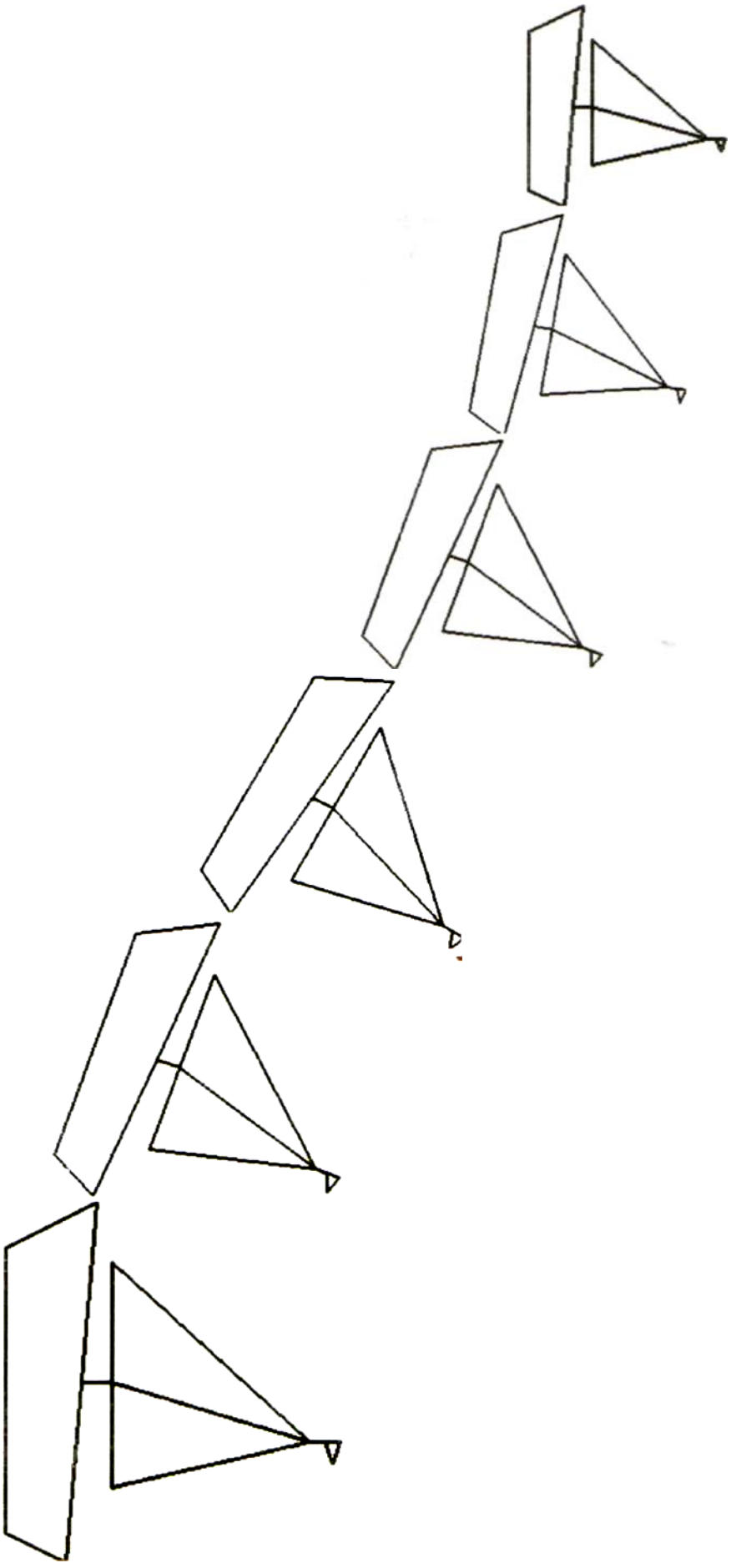
Таким образом, в основу блока геометрических преобразований положена процедура перемножения матриц. Ниже приводится пример такой процедуры, составленной на языке TURBO PASCAL.

```
procedure multiplication_mas(var mas:point;mas1:matrix);
var i:integer;
        x,y,z:real;
begin
        for i:=1 to NYZ do
        begin
{                                   ПЕРЕМНОЖЕНИЕ МАТРИЦ                                  }
{                          NYZ-КОЛИЧЕСТВО УЗЛОВ МАССИВА РИСУНКА                          }
{МАТРИЦА КООРДИНАТ ТОЧЕК ‘MAS’ [NYZ,3] ПЕРЕМНОЖАЕТСЯ НА МАТРИЦУ}
{                                   ПРЕОБРАЗОВАНИЯ ‘MAS1’ [3.3]                          }
x:=(mas[i,1]*mas1[1,1])+ (mas[i,2]*mas1[2,1])+ (mas[i,3]*mas1[3,1]);
y:=(mas[i,1]*mas1[1,2])+ (mas[i,2]*mas1[2,2])+ (mas[i,3]*mas1[3,2]);
z:=(mas[i,1]*mas1[1,3])+ (mas[i,2]*mas1[2,3])+ (mas[i,3]*mas1[3,3]);
mas[i,1]:=x;
mas[i,2]:=y;
mas[i,3]:=z;
        end;
end;
```

## Примеры преобразований





## Листинг программы

```
#include <graphics.h>       // подключение необходимых библиотек
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#defme DIMX 12              // размерности матриц
#define DIMY 3              // постоянная перемещения //
#defmeSDVIG 10              постоянные масштабирования
#define PLUS 1.05
#define MINUS 0.95
#deflne a -3.14/18          // постоянная поворота - 10 градусов
double matkoor [DIMX] [DIMY],     // матрица коор-т об' екта
          center[3];              // матрица коор-т центра об'екта
int       InitMatrKoor();       // ф-ция иниц-ции матрицы коор-т
void      Actions(int choice),// ф-ция действий над об'ектом
                  // choice - код нажатой клавиши
          InitCenter(),           // ф-ция иниц-ции матрицы коор-т центра об'екта
         Motion(int X,int Y,double m[DIM_X][DIM_Y]), // ф-ция смещения
                  //X-смещ-ние по оси X,У -смещ-ние по оси У, m-мат-ца коор-т
         Rotation(double m[DIM_X][DIM_Y],double rad), // ф-ция поворота
                  // m-мат-ца коор-т,rad-угол поворота
         Scaling(double,double,double m[DIM_X] [DIMYiy/ф-ция масшт-ния
         Multiplying(double m[DIM_X][DIM_Y],double m2[DIM_X][DIM_Y]),
         // ф-ция умнож-я матриц   (m на m2 с рез-ом в m)
Painter(double m[DIM_X][DIM_Y]); // ф-ция создания изоб-ния; int
i_see=0; int main( void) {
int gdrawiver = DETECT, gmode»errorcode; //запрос на автообнаружение
initgraph(&gdrawiver, &gmode, "..//bgi");     // инициализация графического ре-
жима
errorcode = graphresult();                    //чтение результата иниц-ии
  if (errorcode != grOk) {                          // ошибка произошла
       printf("Graphics error: %s\n",
       grapherrormsg(errorcode)); printf("Press any key to
       halt:");getch(); exit(l);
        if(InitMatrKoor()==0) exП(1);//если невожможно открыть файл -
        exit for(;;){
          cleardevice();              //очистка экрана
          InitCenter();
          Painter(matkoor);           //вывод изоб-ния
          Actions(getch());           //действия
          closegraph();
            return (0);
         } intInitMatrKoor(){
```

```
   char string[80]; // имя файла с коор-ми
   int coor,ij;
   FILE *in;          // указатель на файл
        printf("Enter filename with coordinates:");gets(string);
        if ((in = fopen(string, "rt")) = NULL){
        fprintf(stderr, "Cannot open input file.\n");
        return 0;
  for( i=O;i<DIM_X;i++)
  for(j=OJ<DIM_Y-1j++){fscanf(in,"%d",&coor);mat_koor[i][ =coor;}
  fclose(in);
  for(i=0;i<DIM_X;i++) mat_koor[i][2]=l;
  return 1;
  >id Painter(double m[DIM X][DIM Y]){
  line( m[0][0], m[0][1], m[l][0],m[1][1]);
  line(m[l][0], m[l][1], m[2][0],m[2][1]);
  line( m[2][0], m[2][1], m[3][0],m[3][1]);
  line( m[3][0], m[3][1], m[4][0],m[4][1]);
  line( m[4][0], m[4][1], m[0][0],m[0][l]);
  line( m[0][0], m[0][1], m[6][0],m[6][1]);
  line(m[5][0],m[5][1], m[7][0],m[7][1]);
  line(m[5][0],m[5][1], m[8][0],m[8][1]);
  line(m[6][0],m[6][1], m[8][0],m[8][1];
  line(m[7][0],m[7][1], m[8][0], m[8][1]);
  line( m[8][0], m[8][1], m[9][0],m m[9][1]);
  line(m[9][0],m[9][1], m[10][0],m[10][l]);
  line(m[10][0],m[10][l],m[11][0],m[11][1]);
  line(m[8][0],m[8]][1], m[11][0], m[11][1]);
    void Actions(int
choice)} switch(choice) {
        case 72 :Motion(0,-SDVIG,mat_koor);break;    //вверх- клавиша
  «UP» case 80 :Motion(0, SDVIG,mat_koor);break;    //вниз- клавиша
  «DOWN» case 75 :Motion(-SDVIG,0,mat_koor);break;//BnpaBO- клавиша
  «RIGHT» case 77 :Motion( SDVIG,0,mat_koor);break;   //влево- клавиша
  «LEFT» case 71 :Motion(-SDVIG,-SDVIG,mat_koor);break;//BBepx и
  влево-
                                                    //клавиша «HOME»
        case 73 :Motion(SDVIG,-SDVIG,mat_koor);break; // вниз и
  влево«END» case 79 :Motion(-SDVIG, SDVIG,mat_koor);break;//BBepx и
  вправо-
                                                    //клавиша «PGUP»
   case 81:Motion(SDVIG,SDVIG,mat_koor);break;//BHH3 и Bnpaeo«PGDN»
  case 43 :Scaling (PLUS, PLUS, mat_koor);breaK;//увеличение-
  «GRAY+»
  case 45 :Scaling(MINUS,MINUS,rnat_koor);break; //уменьшение - «-»
```

```
        case 47 : // поворот по часовой -клавиша «GRAY *»
        Motion(-center[0],-center[l], таг_коог);//перенос в центр коор-т
        Rotation(mat_koor, a);                          //поворот
         Motion(center[0],center[l], matkoor);// перенос в прежнее положение
                i_see++;if(i_see==36) i_see=0/отслеживаем общий угол пов-та
                break; case 42 :// поворот против часовой -клавиша
        «GRAY /»
                Motion(-center[0],-center [I ], matkoor);
                Rotation(mat_koor, -a);
                Motion(center[0],center[l ], matkoor);
                i_see--;if(i_see=-18) i_see= 18;
                break;
        case 119: Scaling(l,PLUS,mat_koor);breaк;//растяжение по оси X- «W»
        case 101:Scaling(MINUS,l,rnat_koor);break; // сжатие по оси X - «E»
        case 113:Scaling(l,PLUS,mat_koor);break; //растяжение по оси Y- «Q»
        case 97:Scaling(l,MINUS,mat_koor);break; // сжатие по оси Y- «A»
        case 27:exit(l);                        //выход из программы - клавиша «ESC»
        default:break;
    }

void InitCenter(){ center [0]=mat_koor [0] [0] ;center [ 1 ]=mat_koor [0][2]
  ;center[2]=mat_koor [0] [2];

void Motion(int X,int Y, double m[DIM_X][DIM_Y]) {
double perenos[DIM_Y][DIM_Y]={l, 0, 0,
                                        0,1,0,
                                        X, Y, 1};    // матрица переноса
    Multiplying( m, регепо5);//умнож-ние мат-цы коор-т на матрицу переноса

void Rotation(double m[DIM_X][DIM_Y], double rad){
  double rotate[DIM_Y][DIM_Y]={ cos(rad),sin(rad),0,
                                        -sin(rad),cos(rad),0,
                                            0,      0,      1 };// матрица поворота
     Multiplying( m, rotate); //умнож-ние мат-цы коор-т на матрицу поворота
void Scaling(double MM1,double MM2, double m[DIM_X][DIM_Y]){
double scale[DIM_Y][DIM_Y]={MMl,    0,    0,
                                        0,     MM2,0,
                                          0,       0,    1 }; //мат-ца масштабирования
        Motion(-center[0],-center[l], m); //перенос в центр коор-т
        Rotation(mat_koor, -a*i_see); //поворот об'екта в исходное положение
        Multiplying (m,sca1e);//умнож-ние мат-цы коор-т на мат-цу масшт-ния
        Rotation(mat_koor, a*i_see);   // поворот об'екта в прежнее положение
        Motion( center[0], center[l], m); // перенос в прежнее положение
```

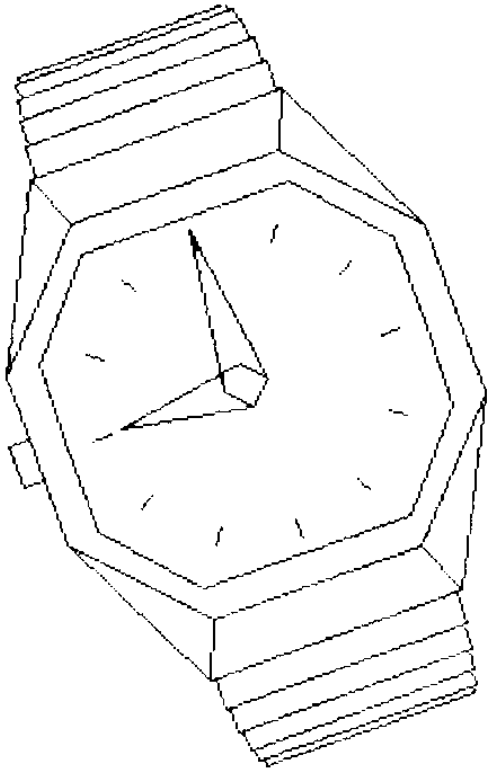
```
void Multiplying(double m[DIM_X][DIM_Y],double m2[DIM_X][DIM_Y]){
double mat_koor2 [DIMX] [DIMY];   //матрица обмена int comp_row,row,col;
//инициализация матрицы обмена
for(row=0;row<DIM_X;row++)for(col=0;col<DIM_Y;col++)
        mat_koor2[row][col]=0;
//непосредственное перемножение матриц m и m2 с рез-том в mat koor2
  for(row=0;row<DIM_X;row++)
     for(comp_row=0;comp_row<DIM_Y;comp_row++)
         for(col=0;col<DIM_Y;col++)
          mat_koor2[row][comp_row] += m[row][col]*m2[col][comp_row]; //обмен
содержимого матриц m и mat_koor2
          for(row=0;row<DIM_X;row++)
      for(col=0;col<DIM_Y;col++)m[row] [col]=rnat_koor2 [row] [col];
```

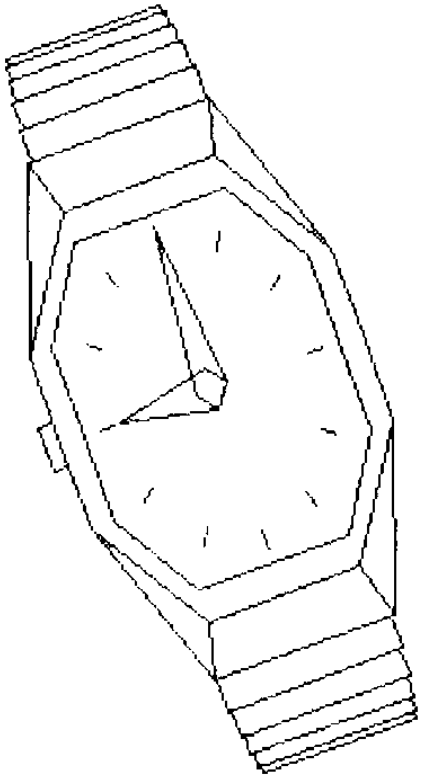
// знач-ния коор-т (поместите в какой-нибудь файл)

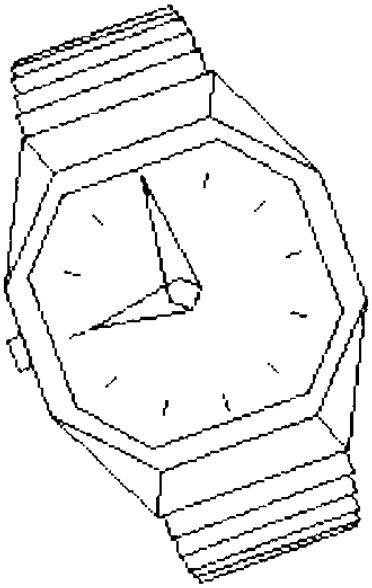
320 240
440 250
420 290
230 290
200 230
390 220
320 220
240 220
360 90
360 80
375 75
360 70

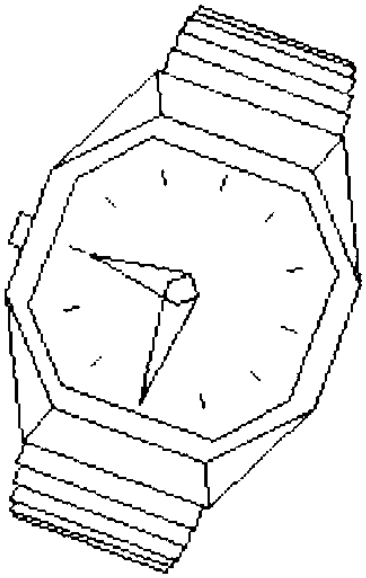
# Варианты заданий

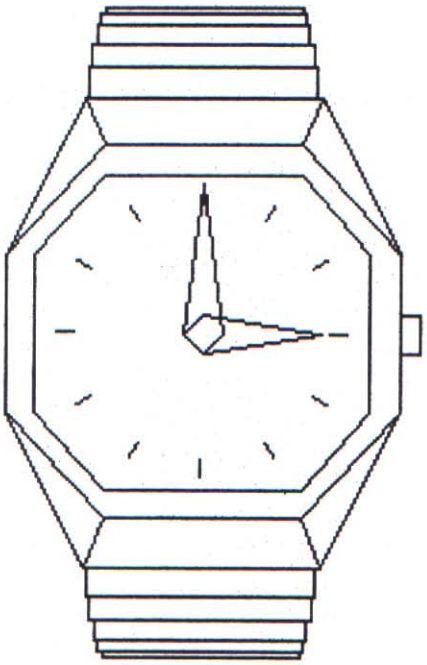
## Вариант 1

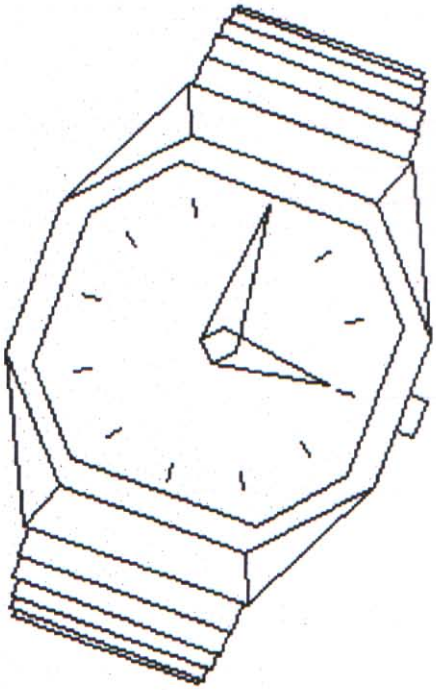


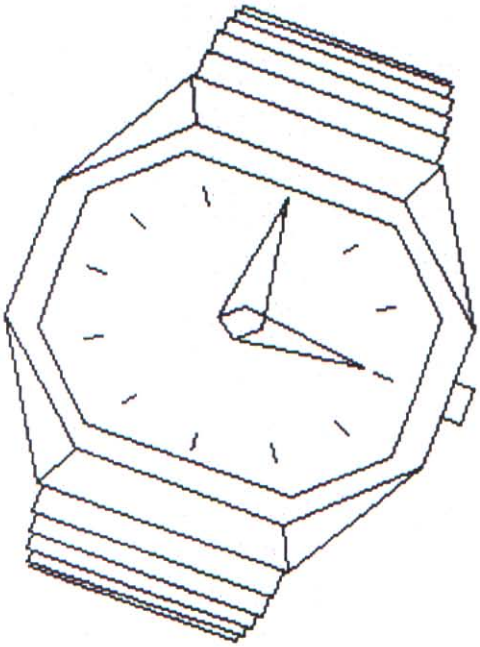


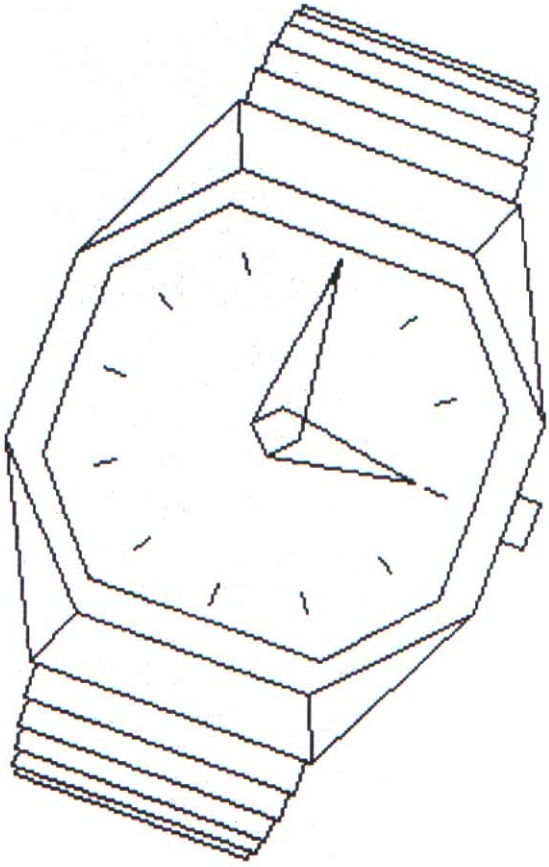




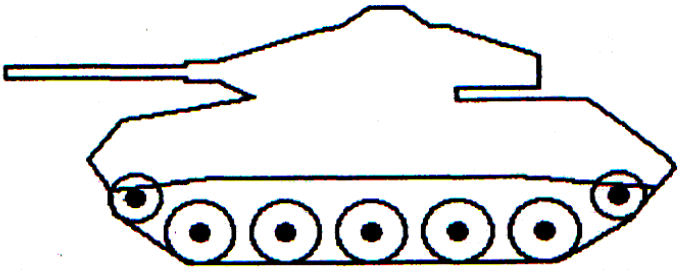


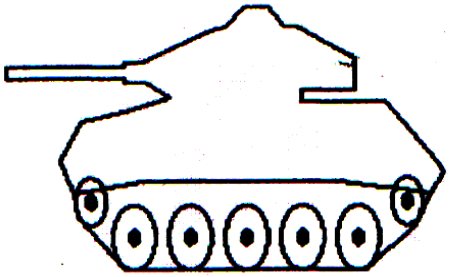


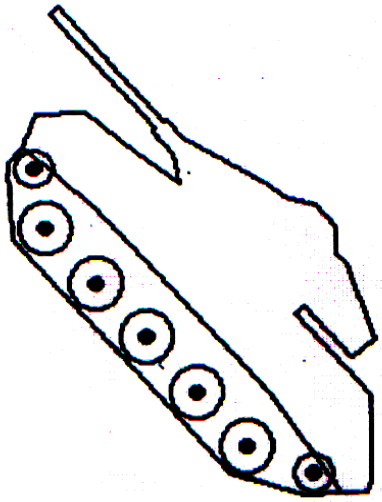


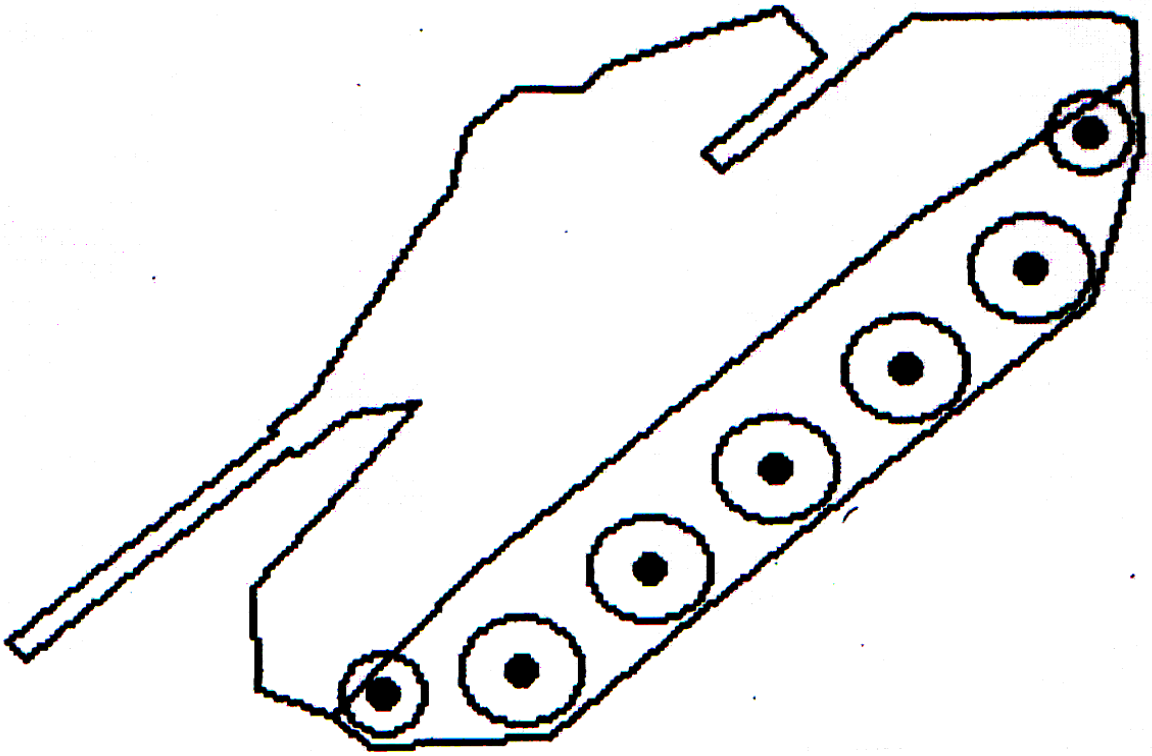


Вариант 2

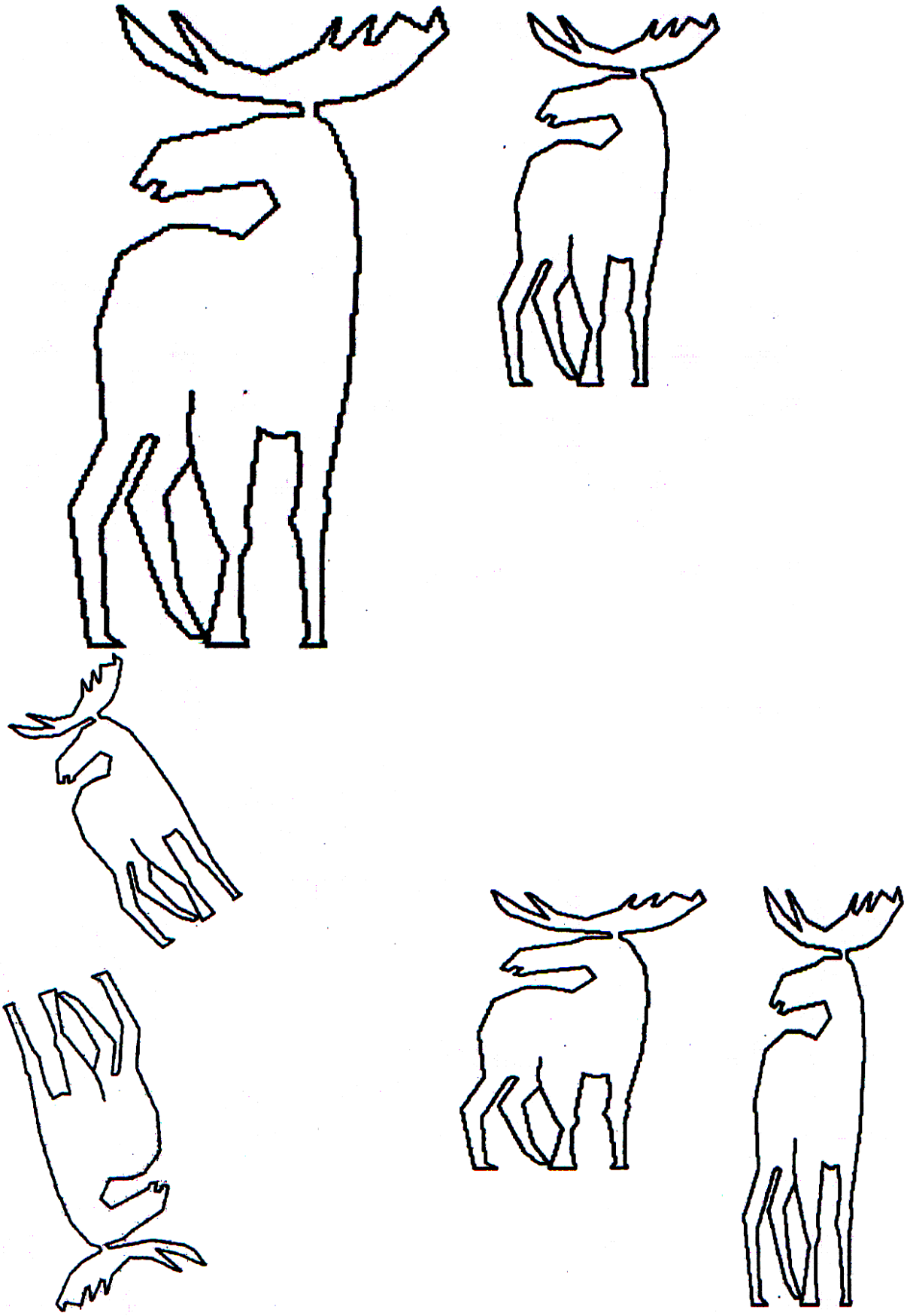




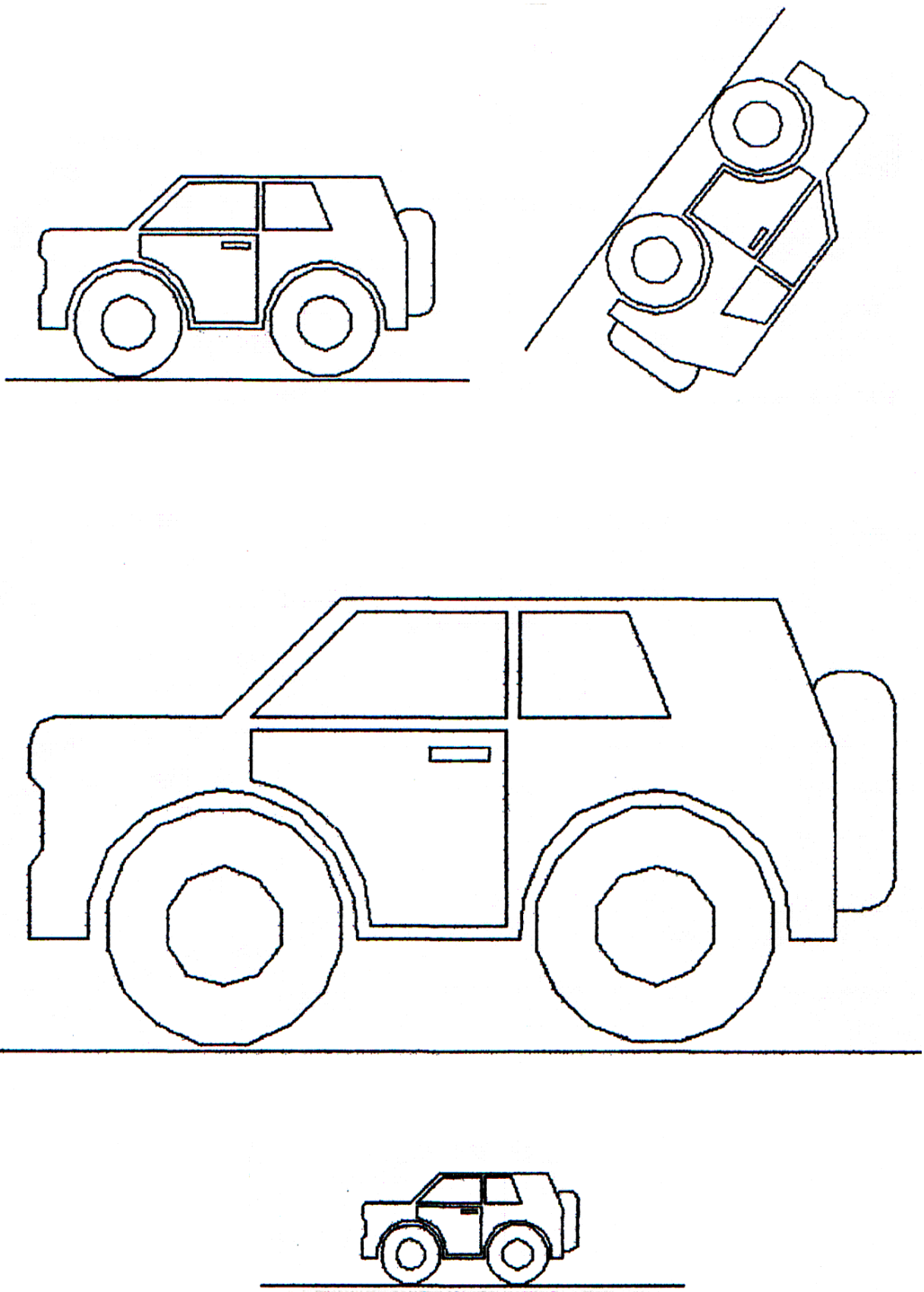




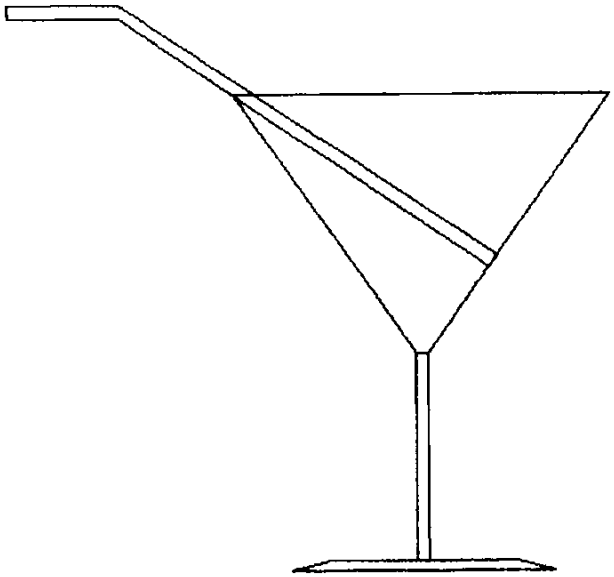
Вариант 3

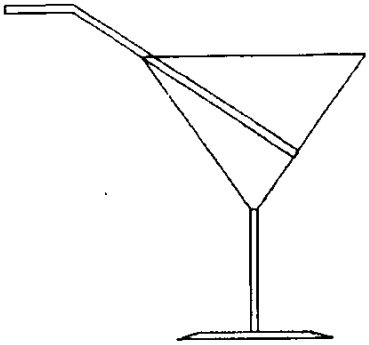


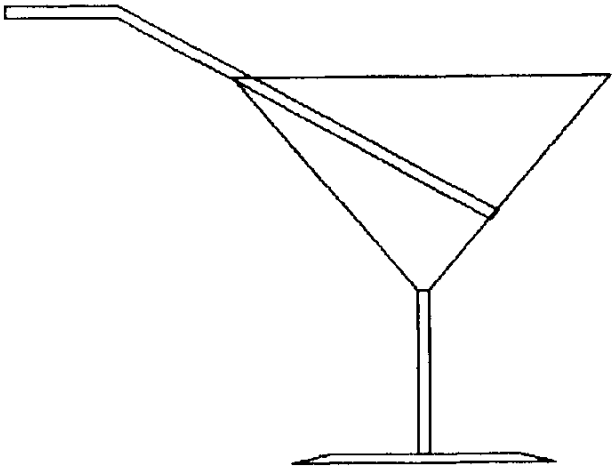
Вариант 4

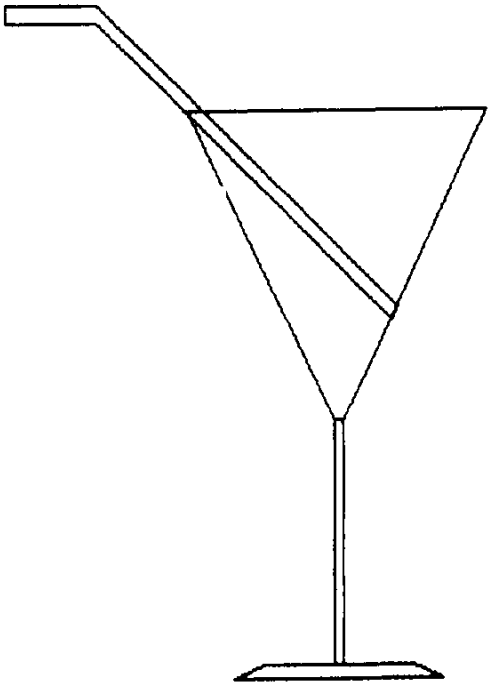


## Вариант 5

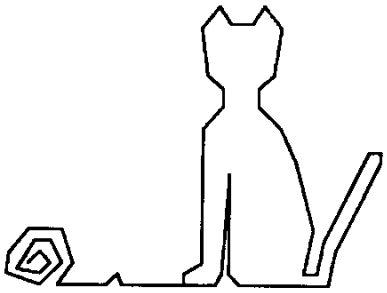


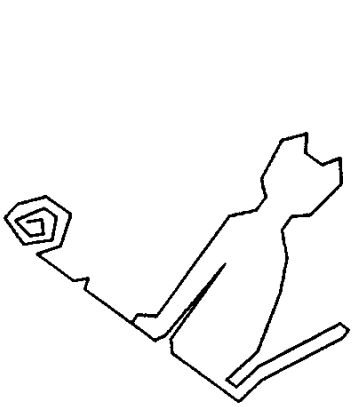


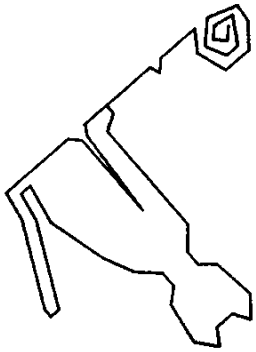


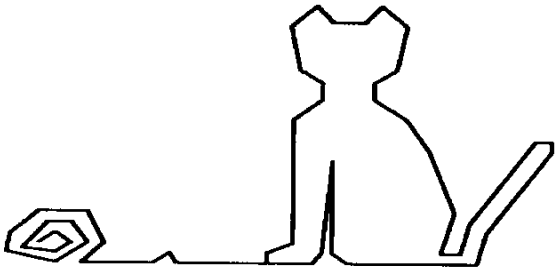


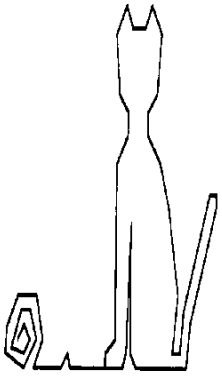
## Вариант 6

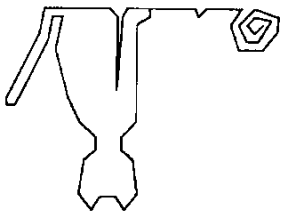


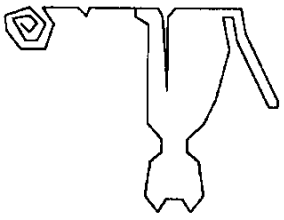


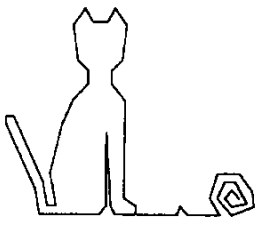












## Оформление отчета по лабораторной работе

В отчете должны быть представлены результаты всех этапов лабораторной работы:

1. Рисунок с указанием опорных точек объекта и их координат.
2. Краткое описание выполняемых геометрических преобразований в матричной форме.
3. Листинг программы, реализующей геометрические преобразования объекта.
4. Твердая копия графического экрана с результатом геометрических преобразований.