# Лабораторная работа №3

## Графические примитивы. Стили линий.

Прежде чем строить графические объекты, нужно понимать какими линиями они будут начерчены. Система КОМПАС предлагает несколько системных стилей линий, каждый из них определяется целым числом. В таблице ниже представлены стили линий

Таблица 1. Стили линий КОСПАСа

| Номер | Описание |
|---|---|
| 1 | Основная линия |
| 2 | Тонкая линия |
| 3 | Осевая линия |
| 4 | Штриховая линия |
| 5 | Линия обрыва |
| 6 | Вспомогательная линия |
| 7 | Утолщенная линия |
| 8 | Штрихпунктирная линия с двумя точками |
| 9 | Штриховая основная |
| 10 | Осевая основная |
| 11 | Тонкая линия, включаемая в штриховку |
| 12 | ISO 02 штриховая линия |
| 13 | ISO 03 штриховая линия (длинный пробел) |
| 14 | ISO 04 штрихпунктирная линия длинный штрих |
| 15 | ISO 05 штрихпунктирная линия длинный штрих 2 пунктира |
| 16 | ISO 06  штрихпунктирная линия длинный штрих 3 пунктира |
| 17 | ISO 07 пунктирная линия |
| 18 | ISO 08 штрихпунктирная линия (длинные и короткие штрихи) |
| 19 | ISO 09 штрихпунктирная линия длинный и 2 коротких штриха |
| 20 | ISO 10 штрихпунктирная линия |
| 21 | ISO 11 штрихпунктирная линия (2 штриха) |
| 22 | ISO 12 штрихпунктирная линия (2 пунктира) |
| 23 | ISO 13 штрихпунктирная линия (3 пунктира) |
| 24 | ISO 14 штрихпунктирная линия (2 штриха 2 пунктира) |
| 25 | ISO 15 штрихпунктирная линия (2 штриха 3 пунктира) |

## Точка

Для построения точки используется метод ksPoint(double x, double y, int style) интерфейса ksDocument2D. Как видно у данного метода всего три параметра: координаты точки и стиль отображения. Стили отображения точки представлены в таблице ниже.

Таблица 2. Стили отображения точки

| Номер | Описание |
|---|---|
| 0 | Точка |
| 1 | Плюс |
| 2 | Буквой X |
| 3 | Квадрат |
| 4 | Треугольник |
| 5 | Окружность |
| 6 | Звезда |
| 7 | Перечеркнутый квадрат |
| 8 | Утолщенный плюс |

Пример использования метода ksPoint() представлен ниже:

```
doc_2d.ksPoint(150, 150, 0);
```

## Прямая

Для построения прямой линии используется метод ksLine(double x, double y, double angle) интерфейса ksDocument2D. У данного метода также три параметра: координаты точки через которую проходит прямая, и угол между прямой и горизонтальной линией. Ниже приведен пример использования данной функции:

```
doc_2d.ksLine(150, 150, 0);
```

## Отрезок

Для построения отрезка используется метод ksLineSeg(double x1, y1, x2, y2, int style) интерфейса ksDocument2D. У данной функции 5 параметров: координаты первой точки отрезка (x1, y1), координаты второй точки отрезка (x2, y2) и стиль линии отрезка. Пример использования данной функции представлен ниже:

```
doc_2d.ksLineSeg(100, 250, 150, 250, 1);
```

## Задание 1

Напишите программу, которая будет создавать документ типа Чертеж конструкторский с ПЗ. Первый лист. ГОСТ 2.104-2006 (101), формат листа А4, укажите имя автора, также назовите данный файл «Лабораторная работа 3 Задание 1 Чертеж 1», ориентация листа вертикальная. В данном чертеже заполните рамку и выведете на чертеж все виды точек с их названием, сохраните данный документ и закройте его. Далее создайте новый чертеж. Отличия второго чертежа от первого: формат А3, название документа «Лабораторная работа 3 Задание 1 Чертеж 2. В данном документе также заполните рамку и выведите на чертеж все типы линий и их названия, сохраните и закройте документ. После проделанных операций нужно

прекратить работу экземпляра КОМПАСа. Пример того, как должны выглядеть чертежи показан на рисунках 1-2.

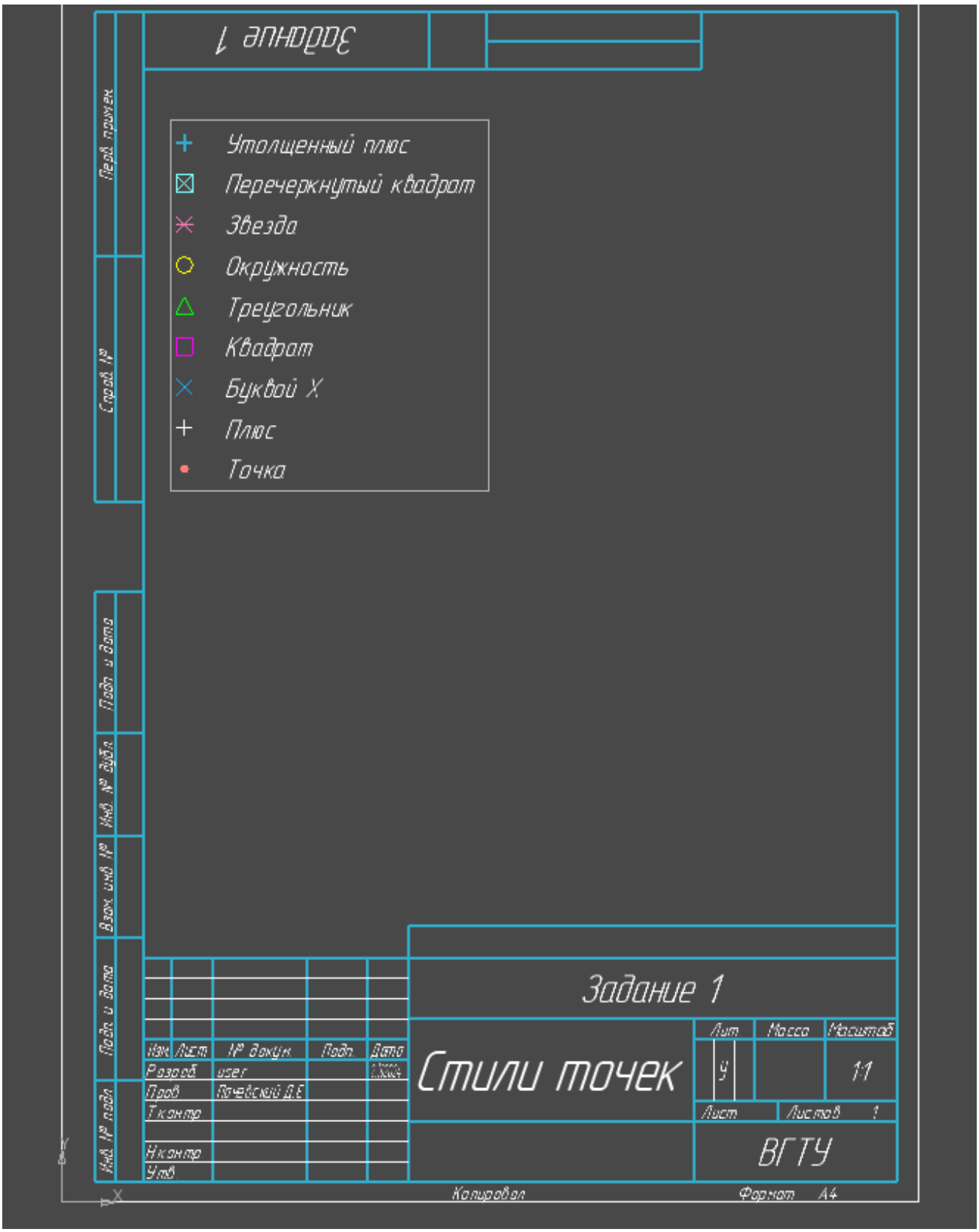



Рисунок 1 – Пример чертежа «Стили точек»

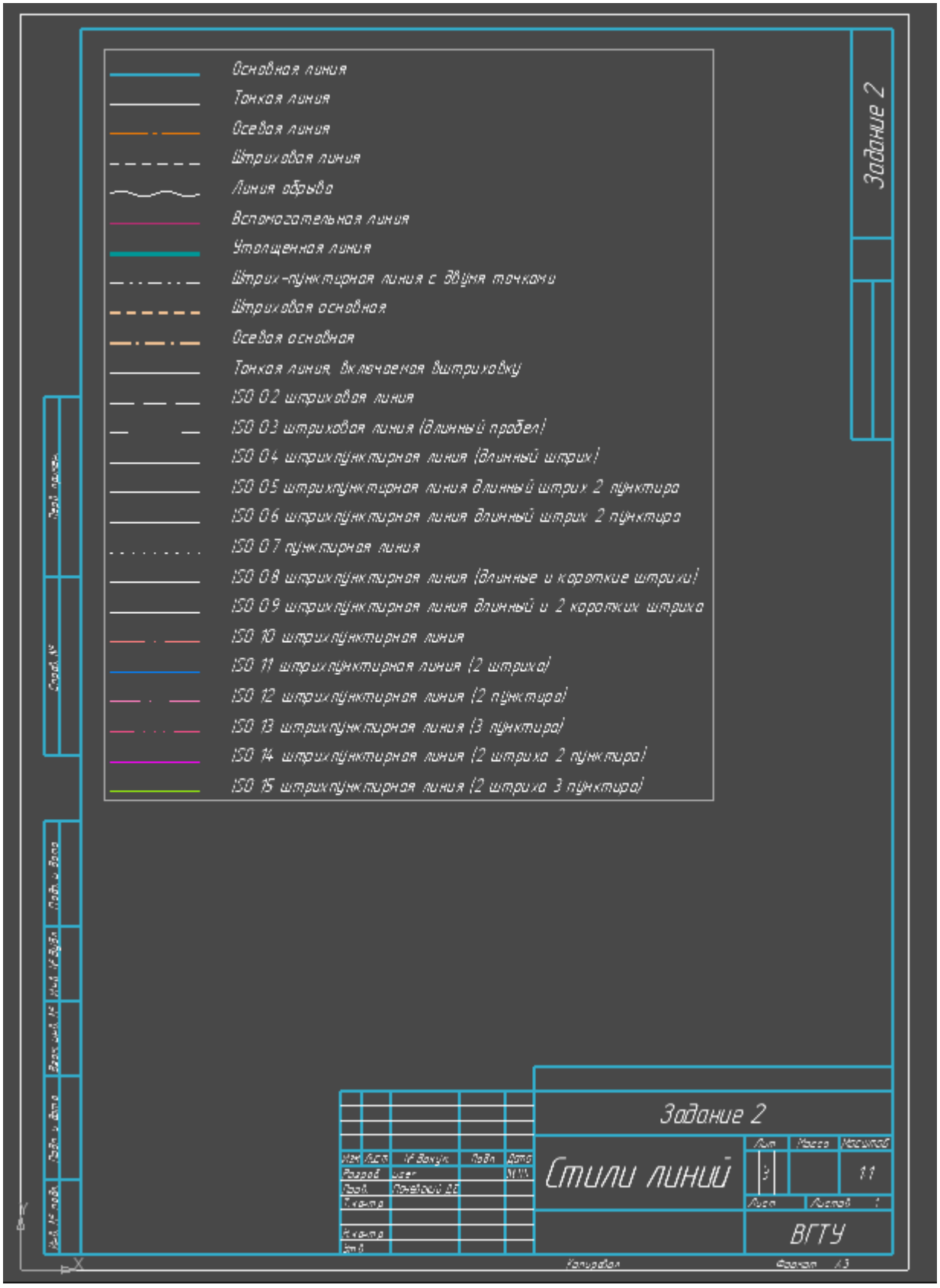



Рисунок 2 – Пример чертежа «Стили линий»

## Окружность

Для построения окружности используется метод ksCircle(double xc, yc, rad; int style) интерфейса ksDocument2D. Ниже представлена реализация данного метода:

```
doc_2d.ksCircle(150, 150, 25, 1);
```

Данный метод принимает 4 аргумента:

1) Xc, yc – координаты центра окружности;
2) Rad – радиус окружности;
3) Style – стиль линии.

## Дуга окружности

Существует три метода построения дуг окружности. Для наиболее лучшей демонстрации данных метод будем использовать их для построения одной и той же дуги. В качестве примера дуги возьмем полуокружность радиусом 10.

Для построения дуги окружности по трем точкам используется метод ksArcBy3Points(double x1, y1, x2, y2, x3, y3 , style) интерфейса ksDocument2D. Ниже представлен пример использования данного метода:

```
doc_2d.ksArcBy3Points(100, 200, 110, 210, 120, 200, 1);
doc_2d.ksText(130, 202.5, 0, 5, 0, 0, "Построение дуги по 3-м точкам");
```

Данный метод принимает семь аргументов:

1) X1, y1 – координаты первой точки.
2) X2, y2 – координаты второй точки.
3) X3, y3 – координаты третьей точки.
4) Style – стиль линии.

Для построения дуги по центру и углам используется метод ksArcByAngle(double xc, yc, rad, f1, f2; int direction, style) интерфейса ksDocument2D. Ниже представлен пример реализации данного метода:

```
doc_2d.ksArcByAngle(110, 180, 10, 0, 180, 1, 1);
doc_2d.ksText(130, 182.5, 0, 5, 0, 0, "Построение дуги по центру и углам");
```

Данный метод принимает следующие аргументы:

1) Xc, yc – координаты центра дуги.
2) Rad – радиус дуги.
3) F1, f2 – углы дуги.
4) Direction – направление дуги.
5) Style – стиль линии.

Стоит обратить внимание, что построение ведется от угла f1 к углу f2. Параметр direction задает направление, в котором следует отрисовывать дугу, если данный параметр равен 1, то дуга отрисовывается против часовой стрелки, иначе по часовой.

Для построения дуги по центру и конечным точкам используется метод ksArcByPoint(double xc, yc, rad, x1, y1, x2, y2; int direction, style) интерфейса ksDocument2D. Ниже представлен пример использования данного метода:

```
doc_2d.ksArcByPoint(110, 160, 10, 100, 160, 120, 160, -1, 1);
     doc_2d.ksText(130, 162.5, 0, 5, 0, 0, "Построение дуги по трем
точкам");
```

Данный метод принимает следующие аргументы:

1) Xc, yc – координаты центра дуги.
2) Rad – радиус дуги.
3) X1, y1 – координаты первой точки.
4) X2, y2 – координаты второй точки.
5) Direction - направление отрисовки.
6) Style – стиль линии.

Данный метод очень похож на метод ksArcByAngle(). Разница между ними только в том, что конечные точки задаются не углами, а координатами.

## Прямоугольник. Параметры прямоугольника

Параметры прямоугольника задаются интерфейсом ksRectangleParam. Получить данный метод можно при помощи метода GetParamStruct() интерфейса KompasObject, для этого надо в качестве параметра передать ему константу ko_RectangleParam. Ниже представлены свойства данного интерфейса:

1) Ang – угол наклона прямоугольника к горизонтально оси;
2) Height – высота прямоугольника.
3) Width – ширина прямоугольника.
4) Style – стиль линии.
5) X и y – координаты левого нижнего угла прямоугольника.

Для построения прямоугольника используется метод ksRectangle(const ksRectangleParam rec_param; int centre) интерфейса ksDocument2D. Ниже представлен фрагмент программы, которая строит прямоуголник:

```
ksRectangleParam rectangle_par;
rectangle_par = (ksRectangleParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RectangleParam);
rectangle_par.ang = 0;
rectangle_par.height = 20;
```

```
rectangle_par.width = 30;
rectangle_par.style = 1;
rectangle_par.x = 100;
rectangle_par.y = 240;
doc_2d.ksRectangle(rectangle_par, 0);
     doc_2d.ksText(140, 247.5, 0, 5, 0, 0, "Никак не обозначать
центр");
```

Центр прямоугольника можно обозначать 4-мя различными способами. В таблице ниже представлен переменные и описание способа для обозначения центра прямоугольника.

Таблица.3. Обозначение центра прямоугольника.

| **Значение** | Описание |
|---|---|
| 0 | Никак не обозначать |
| 1 | Маленький крестик |
| 2 | Горизонтальная ось |
| 3 | Горизонтальная и вертикальная оси |

## Правильный многоугольник

Параметры правильного многоугольника задаются интерфейсом ksRegularPolygonParam. Получить данный интерфейс можно с помощью GetParamStruct() интерфейса KompasObject, для этого ему в качестве параметра нужно передать константу ko_RegularPolygonParam. Ниже представлены свойства данного интерфейса:

1) Ang – угол между вертикальной осью и одной из вершин многоугольника.
2) Count – количество вершин.
3) Describe – определяет значение свойства radius. Если свойство describe имеет значение true, тогда свойство radius рассматривается как радиус вписанной в многоугольник окружности. В противном случае свойство radius рассматривается как радиус описанной вокруг многоугольника окружности.
4) Radius – радиус вписанной или описанной окружности.
5) Style – стиль линий многоугольника.
6) Xc и yc – координаты центра вписанной или описанной окружности.

Для построения многоугольника используется метод ksRegularPolygon интерфейса ksDocument2D. Ниже представлен фрагмент программы, которая строит прямоугольник:

```
ksRegularPolygonParam reg_pol_par;
reg_pol_par = (ksRegularPolygonParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RegularPolygonParam);
reg_pol_par.ang = 45;
reg_pol_par.count = 8;
```

```
reg_pol_par.radius = 30;
reg_pol_par.style = 1;
reg_pol_par.xc = 150;
reg_pol_par.yc = 150;
doc_2d.ksRegularPolygon(reg_pol_par);
```

### Эллипс. Параметры эллипса

Параметры эллипса описываются интерфейсом ksEllipseParam. Получить его можно при помощи метода GetParamStruct() интерфейса KompasObject для этого в метод нужно передать константу ko_EllipseParam. Ниже представлены свойств этого интерфейса:

1) A, B – размеры полуосей эллипса.
2) Angle – угол наклона эллипса.
3) Style – стиль линии эллипса.
4) Xc, yc – координаты центра эллипса, то есть точки пересечения его осей.

Построить эллипс можно при помощи метода ksEllipse() интерфейса ksDocument2D. Ниже представлен фрагмент программ, которая строит эллипс:

```
ksEllipseParam ellipse_par;
ellipse_par = (ksEllipseParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_EllipseParam);
ellipse_par.A = 40;
ellipse_par.B = 20;
ellipse_par.angle = 0;
ellipse_par.style = 1;
ellipse_par.xc = 150;
ellipse_par.yc = 150;
doc_2d.ksEllipse(ellipse_par);
```

### Дуга эллипса. Параметры дуги эллипса

Параметры дуги эллипса задаются интерфейсом ksEllipseArcParam. Получить данный интерфейс можно при помощи метода GetParamStruct() интерфейса ksDocument2D, передав в метод константу ko_EllipseArcParam. Ниже представлены свойства этого интерфейса:

1) A, B – размеры полуосей эллипса.
2) angleFirst – угол первой точки дуги.
3) Angle – угол наклона оси эллипса.
4) angleSecond – угол второй точки дуги.
5) direction – направление отрисовки дуги.
6) Style – стиль линии дуги.
7) Xc, yc – координаты центра эллипса, то есть точки пересечения его осей.

Для построения дуги эллипса используется метод ksEllipseArc() интерфейса ksDocument2D. Ниже представлен фрагмент программы, которая строит дугу эллипса:

```
ksEllipseArcParam ellipse_arc_par;
ellipse_arc_par = (ksEllipseArcParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_EllipsArcParam);
ellipse_arc_par.A = 40;
ellipse_arc_par.B = 20;
ellipse_arc_par.angleFirst = 0;
ellipse_arc_par.angleSecond = 180;
ellipse_arc_par.angle = 0;
ellipse_arc_par.direction = 1;
ellipse_arc_par.style = 1;
ellipse_arc_par.xc = 150;
ellipse_arc_par.yc = 150;
doc_2d.ksEllipseArc(ellipse_arc_par);
```

### Ломанная

Перед тем как начать построение ломанной линии нужно познакомиться с двумя методами интерфейса ksDocument2D: ksPolyline(int style) и ksEndObj(). Первый метод фактически дает старт построению ломанной линии, а второй метод как раз останавливает данный процесс. Ниже представлен фрагмент программы, который строит ломанную линию:

```
doc_2d.ksPolyline(1);
doc_2d.ksPoint(100, 100, 0);
doc_2d.ksPoint(110, 110, 0);
doc_2d.ksPoint(120, 90, 0);
doc_2d.ksPoint(130, 110, 0);
doc_2d.ksPoint(140, 90, 0);
doc_2d.ksPoint(150, 110, 0);
doc_2d.ksPoint(160, 90, 0);
doc_2d.ksPoint(170, 100, 0);
doc_2d.ksEndObj();
```

### Кривая Безье. Точка кривой Безье

Точки кривой Безье выводятся не так, как обычные точки ( методом ksPoint() ). Это связанос тем, что для построения кривой одних координат точек недостаточно. Каждая точка кривой описывается интерфейсом ksBezierPointParam. Получить данный интерфейс можно с помощью метода GetParamStruct интерфейса KompasObject. Для этого в него нужно передать константу ko_BezierPointParam. Свойства данного интерфейса представлены ниже:

1) Ang – угол наклона касательной к кривой в этой точке.
2) Left – расстояние от базовой точки до левой точки узла.
3) Right – расстояние от базовой точки до правой точки узла.
4) X, y – координаты точки.

Для лучшего понимания назначения параметров left и right рекомендую посмотреть алгоритм построения кривой Безье.

Метод у данного интерфейса всего один – Init(). Он сбрасывает параметры точки.

Для построения точки кривой используется метод ksBezierPoint() интерфейса ksDocument2D.

Процесс построения кривой Безье запускается методом ksBezier(int closed, int style). Ниже представлен фрагмент программы, который строит кривую Безье:

```
ksBezierPointParam bezier_point_par;

bezier_point_par = (ksBezierPointParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_BezierPointParam);
bezier_point_par.Init();

doc_2d.ksBezier(0, 1);

bezier_point_par.x = 100;
bezier_point_par.y = 100;
bezier_point_par.ang = 45;
doc_2d.ksBezierPoint(bezier_point_par);

bezier_point_par.x = 110;
bezier_point_par.y = 110;
bezier_point_par.ang = 0;
doc_2d.ksBezierPoint(bezier_point_par);

bezier_point_par.x = 120;
bezier_point_par.y = 90;
bezier_point_par.ang = 0;
doc_2d.ksBezierPoint(bezier_point_par);

bezier_point_par.x = 130;
bezier_point_par.y = 110;
bezier_point_par.ang = 0;
doc_2d.ksBezierPoint(bezier_point_par);

doc_2d.ksEndObj();
```

### NURBS кривая

Построение NURBS кривой очень похоже на построение ломаной. Разница лишь в том, что запуск построения кривой осуществляется методом ksNurbs(int degree, bool close, style) интерфейса ksDocument2D. Подробно поговорим про аргументы, которые принимает данная функция:

1) Degree – степень полинома кривой (от 3 до 8).
2) Close – признак замкнутой кривой.

3) Style – стиль линии.

Ниже представлен фрагмент программы, которая строит NURBS кривую:

```
doc_2d.ksNurbs(3, false, 1);
doc_2d.ksPoint(100, 100, 0);
doc_2d.ksPoint(110, 110, 0);
doc_2d.ksPoint(120, 90, 0);
doc_2d.ksPoint(130, 110, 0);
doc_2d.ksPoint(140, 90, 0);
doc_2d.ksPoint(150, 110, 0);
doc_2d.ksPoint(160, 90, 0);
doc_2d.ksPoint(170, 100, 0);
doc_2d.ksEndObj();
```

## Симметрия

Предположим, что нам нужно выполнить чертеж вала. Конечно его можно построить из двух прямоугольников или из отрезков. Правильнее конечно, вначале построить верхнюю часть вала, а потом получить нижнюю часть путем симметричного отображения верхней части относительно оси.

Вначале нужно создать группы объектов (отрезков, дуг), над которыми будет выполняться симметричное отображение. Для создания группы используется два метода интерфейса ksDocument2D – ksNewGroup(int type (тип группы)) и ksEndObj(). Всего существует два типа группы: модельная (type = 0) и временная (type = 1). Элементы временной группы в отличие от элементов модельной группы не попадают на чертеж и уничтожаются сразу после завершения какой-либо программы.

После того как мы создали группу объектов, можно приступить к работе с симметричным отображением. Осуществляется это методом ksSymmetryObj(int ref, double x1, y1, x2, y2; string copy) интерфейса ksDocument2D. Ниже представлено описание входных параметров данного метода:

1) Ref – ссылка на группу объектов.
2) X1, y1 – 1-ая точка оси симметрии.
3) X2, y2 – 2-ая точка оси симметрии
4) Copy – режим копирования (определяет как следует поступить с исходными объектами после отображения. Если он равен 0, то исходные объекты будут удалены, если же он равен 1, то исходные объекты останутся без изменения).

Ниже приводится фрагмент программы, которая осуществляет построение вала путем симметричного отображения:

```
int group_id;
group_id = doc_2d.ksNewGroup(0);
doc_2d.ksLineSeg(100, 100, 100, 120, 1);
doc_2d.ksLineSeg(100, 120, 120, 120, 1);
```

```
doc_2d.ksLineSeg(120, 120, 120, 100, 1);
doc_2d.ksLineSeg(120, 110, 160, 110, 1);
doc_2d.ksLineSeg(160, 110, 160, 100, 1);
doc_2d.ksEndObj();

doc_2d.ksSymmetryObj(group_id, 100, 100, 110, 100, "1");
```

## Поворот

Предположим, что перед нами стоит задача построить повернутый равнобедренный треугольник. Можно конечно вспомнить школьный курс геометрии и вычислить координаты вершин этого треугольника после поворота. А можно поступить проще. Сначала построить не повернутый треугольник, а потом повернуть его средствами системы КОМПАС.

Осуществление этой операции очень похоже на осуществление симметричного отображения, о котором мы говорили в прошлой главе. Здесь также как и в случае с симметрией сначала создается группа объектов, которая будет повернута, после чего выполняется сам поворот.

Сам поворот осуществляется методом ksRotateObj(int ref; double x, y, ang) интерфейса ksDocument2D. Ниже представлено описание входных параметром данного метода:

1) Ref – ссылка на группу объектов.
2) X, y – координаты центра поворота.
3) Angle – угол на который нужно повернуть объект.

Ниже представлен фрагмент программы, которая осуществляет поворот объекта:

```
int group_id;

group_id = doc_2d.ksNewGroup(0);

doc_2d.ksLineSeg(100, 100, 120, 110, 1);
doc_2d.ksLineSeg(100, 100, 120, 90, 1);
doc_2d.ksLineSeg(120, 110, 120, 90, 1);
doc_2d.ksEndObj();

doc_2d.ksRotateObj(group_id, 100, 100, 45);
```

## Построение дуги окружности касательной к двум прямым

Предположим, нужно построить дугу окружности, представляющую собой сопряжение двух прямых.

Для решения данной задачи нужно познакомиться с интерфейсом ksCON, который описывает параметры дуг окружностей сопрягающих две прямых. Получить

его можно с помощью метода GetParamStruct() интерфейса KompasObject. Для этого нужно данному методу передать константу ko_CON.

Свойств у интерфейса ksCON нет, у него есть только методы:

1) GetXc, GetYc – возвращает координаты центра сопрягающей дуги.
2) GetX1, GetY1 – возвращают координаты первой точки сопрягающей дуги.
3) Getx2, GetY2 – возвращают координаты второй точки сопрягающей дуги.

За осуществление математических расчетов отвечает интерфейс ksMathematic2D() интерфейса KompasObject. Данный метод не имеет входных параметров. Среди всех свойств и методов данного интерфейса нам интересен всего один метод ksCouplingLineLine(double x1, y1, angle1, x2, y2, angel2, rad; param). Ниже представлено описание входных параметров данного метода:

1) X1, y1 – координаты точки на первой сопрягаемой прямой.
2) Angle1 – угол наклона первой сопрягаемой прямой.
3) X2, y2 – координаты точки на второй сопрягаемой прямой.
4) Angle2 – угол наклона второй сопрягаемой прямой.
5) Rad – радиус сопрягающей дуги окружности.
6) Param – интерфейс ksCON, куда будут записаны результаты.

Ниже приведен фрагмент программы, которая осуществляет расчет сопрягающей дуги окружности и строит данную дугу:

```
ksMathematic2D math_2d;
ksCON con;

math_2d = (ksMathematic2D)kompas.GetMathematic2D();
con = (ksCON)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_CON);

math_2d.ksCouplingLineLine(80, 110, 0,
                           110, 90, 90,
                           10,
                           con);
doc_2d.ksArcByPoint(con.GetXc(1), con.GetYc(1), 10,
                    con.GetX1(1), con.GetY1(1),
                    con.GetX2(1), con.GetY2(1),
                    - 1, 1);
doc_2d.ksLineSeg(80, 110, con.GetX1(1), con.GetY1(1), 1);
     doc_2d.ksLineSeg(110, 90, con.GetX2(1), con.GetY2(1), 1);
```

## Построение прямой касательной к двум окружностям

Для описания точек касания используется интерфейс ksTAN. Получить этот интерфейс можно с помощью метода GetParamStruct интерфейса KompasObject. Для этого последнему в качестве параметра нужно передать константу ko_tan.

Данный интерфейс содержит информацию о двух точках касания прямой с двумя окружностями. Данная информация заключается в координатых соответствующих точек касания. Отсюда и возникают 4 метода млм свойства.

Согласно документации для получения данных точек используется 4 метода GetX1, GetY1, GetX2, GetY2, которые в качестве единственного параметра принимают индекс найденной точки касания.

Для получения координат точек касания используется интерфейс ksMathematic2D. Для получения данного интерфейса используется метод GetMathematic2D интерфейса KompasObject. Среди всех методов интерфейса ksMathematic2D нам интересен метод ksTamCircleCircle(double xc1, yc1, radius1. Xc2, yc2, radius2; param). Именно этот метод отвечает за получение координат точек касания между прямой и двумя окружностями. Ниже Ниже представлено описание входных параметров данного метода:

1) Xc1, yc1 – координаты центра первой окружности.
2) Radius1 – радиус первой окружности.
3) Xc2, yc2 – координаты центра второй окружности.
4) Radius2 – радиус второй окружности.
5) Param – интерфейс ksTAN.

Ниже приводится фрагмент программы, которая строит отрезок и две дуги сопряженный с этим отрезком:

```
ksMathematic2D math_2d;
ksTAN tan;

math_2d = (ksMathematic2D)kompas.GetMathematic2D();
tan = (ksTAN)kompas.GetParamStruct((short)Kompas6Constants.
    StructType2DEnum.ko_TAN);

math_2d.ksTanCircleCircle(100, 100, 10,
                          150, 150, 20,
                          tan);
doc_2d.ksLineSeg(tan.x1[2], tan.y1[2], tan.x2[2], tan.y2[2], 1);
doc_2d.ksArcByPoint(100, 100, 10, tan.x1[2], tan.y1[2], 90, 100, -1, 1);
     doc_2d.ksArcByPoint(150, 150, 20, tan.x2[2], tan.y2[2], 190, 150, -1,
1);
```

## Построение перпендикуляра к отрезку

Предположим, что у нас есть какой-то отрезок. Как построить перпендикуляр к этому отрезку, проходящий через заданную точку?

Если нужно построить перпендикуляр к горизонтальному или вертикальному отрезку, то это не проблема: строим вертикальный или горизонтальный отрезок. А если изначальный отрезок ориентирован произвольный образом. Тогда есть два метода:

1) Решать данную задачу вручную с использованием знаний аналитической геометрии или векторной алгебры.
2) Использовать систему КОМПАС, которая сама решит эту задачу за нас.

У интерфейса ksMathematic2D есть специальный метод, предназначенный как раз для решения нашей задачи ksPerpendicular (double x, y, x1, y1, x2, y2, xp, yp). Ниже приведено описание входных параметров данного метода:

1) X, y – координаты точки, через которую должен проходить перпендикуляр.
2) x1, y1 – координаты  первой точки отрезка, к которому ищется перпендикуляр.
3) X2, y2 – координаты второй точки отрезка, к которому ищется перпендикуляр.
4) Xp, yp – в эти параметры будут записаны координаты найденной точки пересечения отрезка и перпендикуляра к нему.

Ниже приводится фрагмент программы, которая демонстрирует работу с данным методом:

```
ksMathematic2D math_2d;
double xp, yp;

doc_2d.ksLineSeg(100, 100, 120, 140, 1);
math_2d = (ksMathematic2D)kompas.GetMathematic2D();
doc_2d.ksPoint(140, 150, 5);
math_2d.ksPerpendicular(140, 150,
                        100, 100,
                        120, 140,
                        out xp, out yp);
doc_2d.ksLineSeg(140, 150, xp, yp, 1);
doc_2d.ksPoint(xp, yp, 1);
```

**Задание 2.**

Написать программу которая создает чертеж типа «Чертеж конструкторский. Первый лист. ГОСТ 2.104-2006.» Заполнить основную надпись и выполнить чертеж без нанесения размеров в соответствие с выбранным вариантом. Пример созданного чертежа для варианта 18 представлен на рисунке ниже. Также важно, чтобы чертеж назывался также как называется деталь и автором были Вы или Ваше имя пользователя ПК

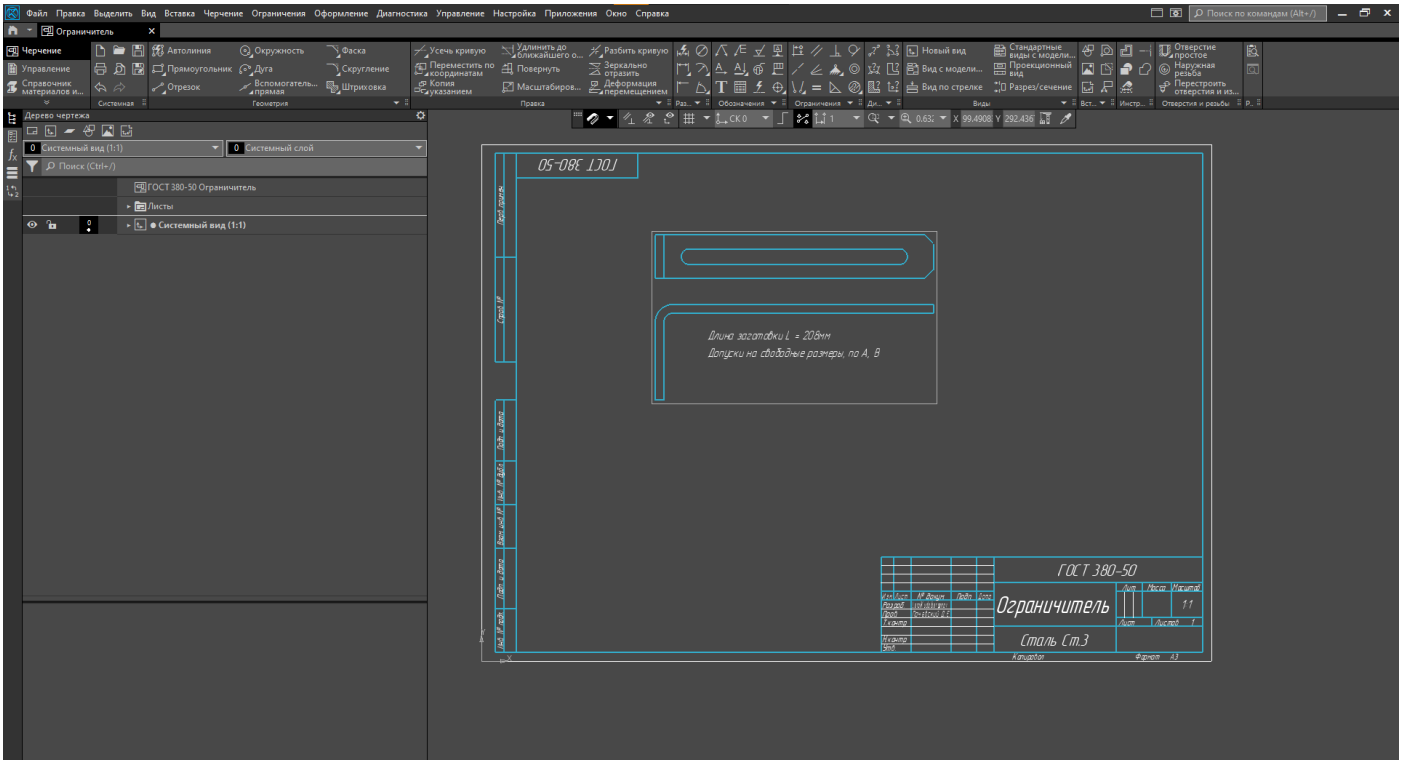
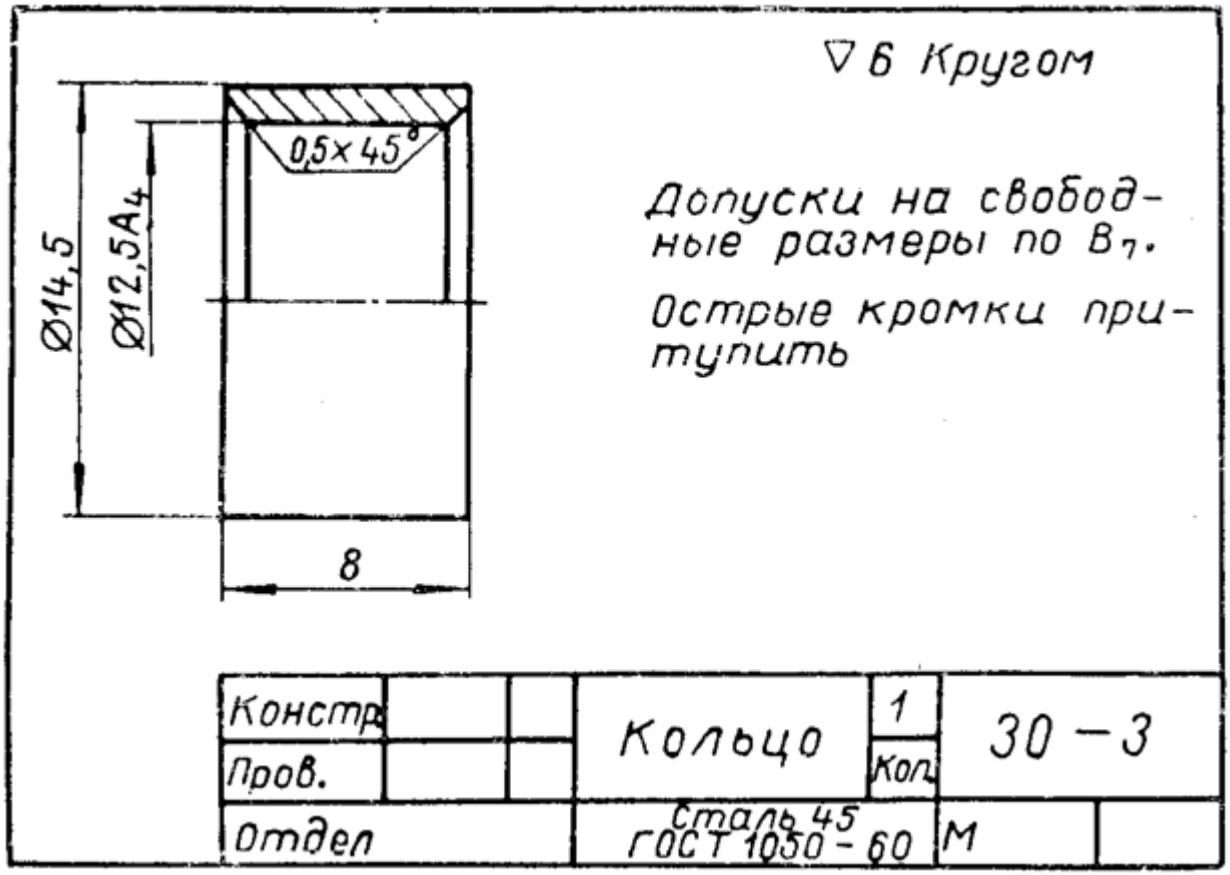
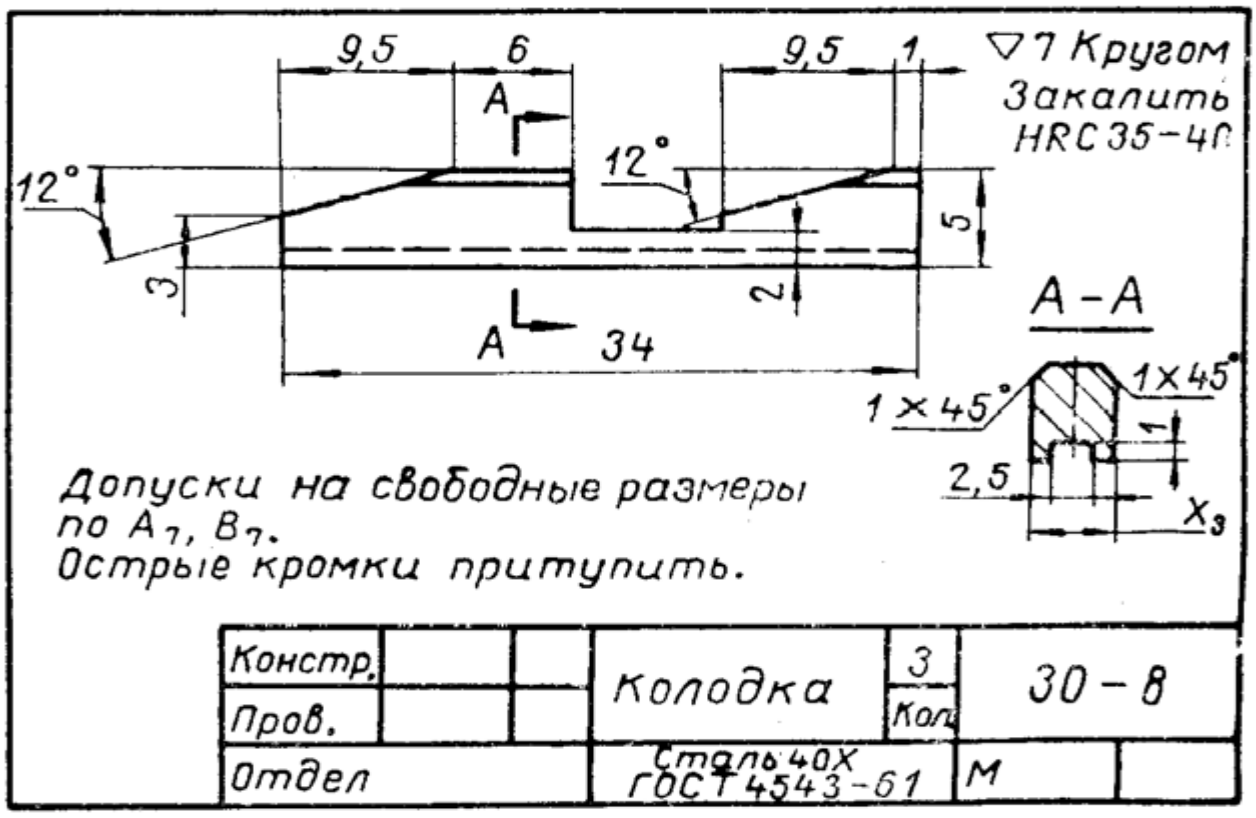



Рисунок 1 – Пример чертежа для варианта 18.

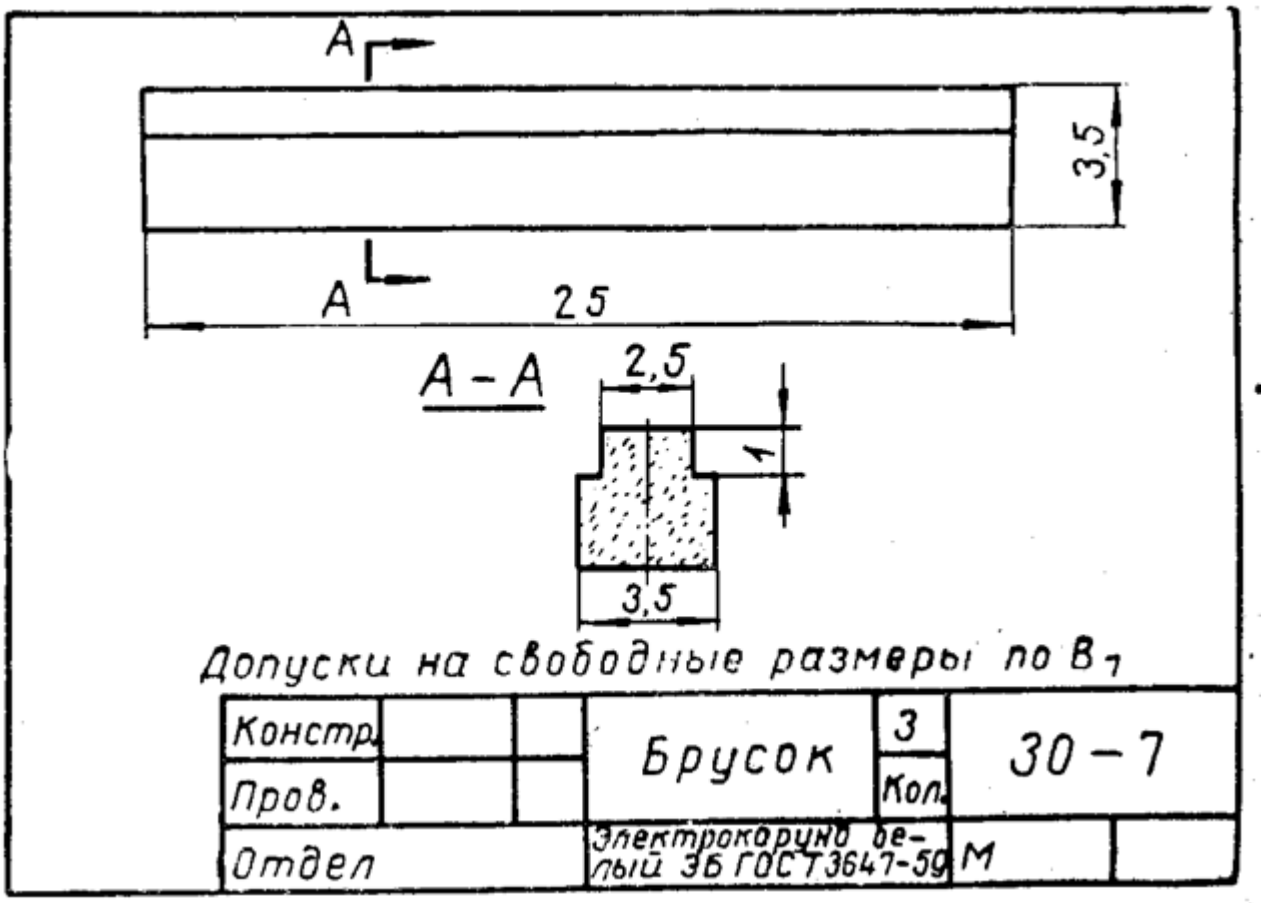
Вариант 1.



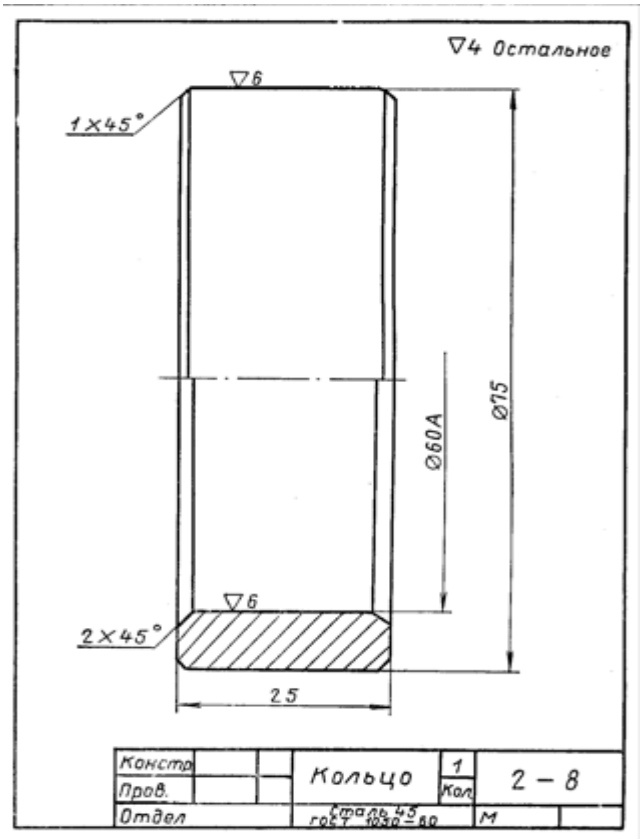

Вариант 2.




**Вариант 3.**

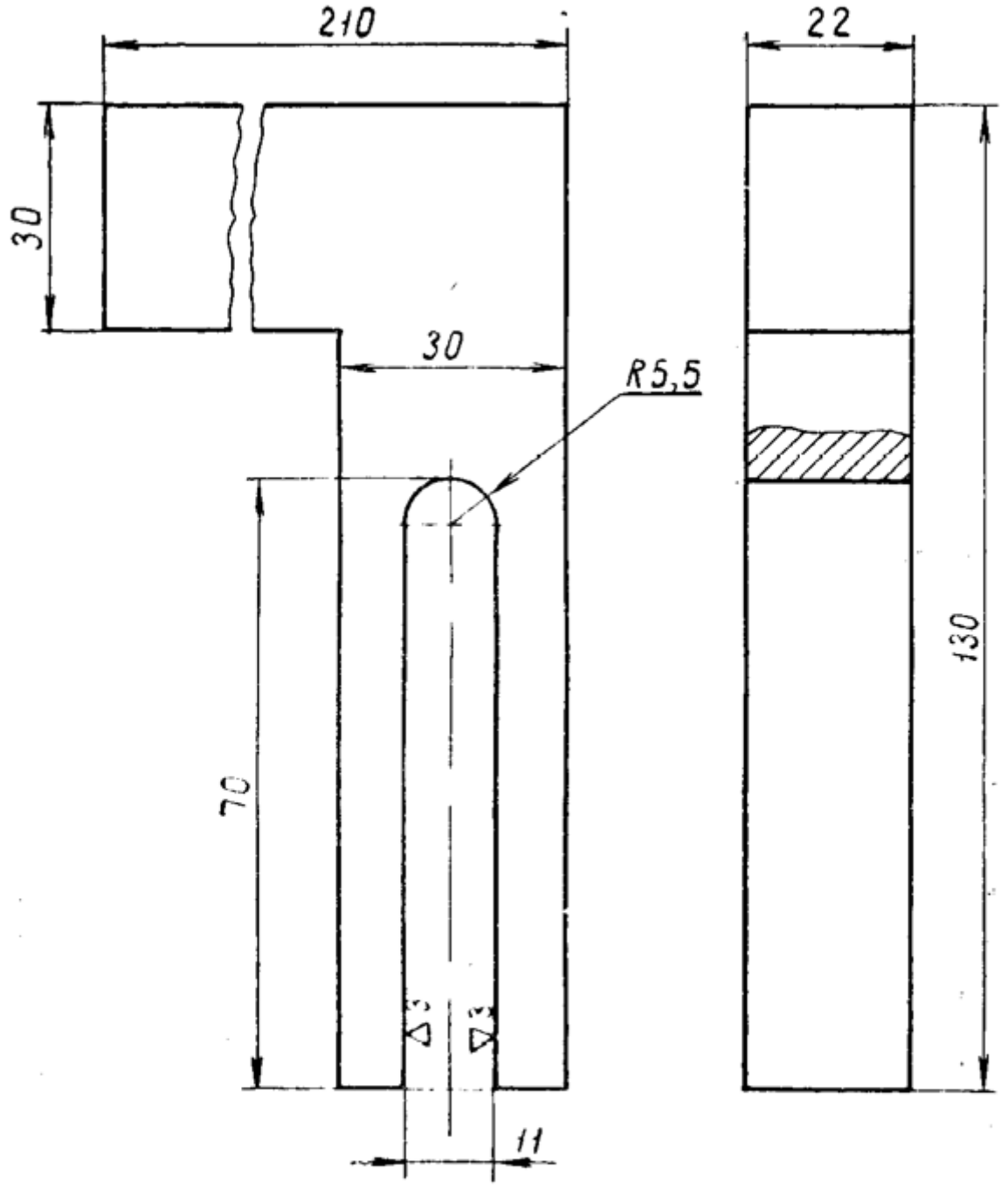



Вариант 4.




**Вариант 5.**

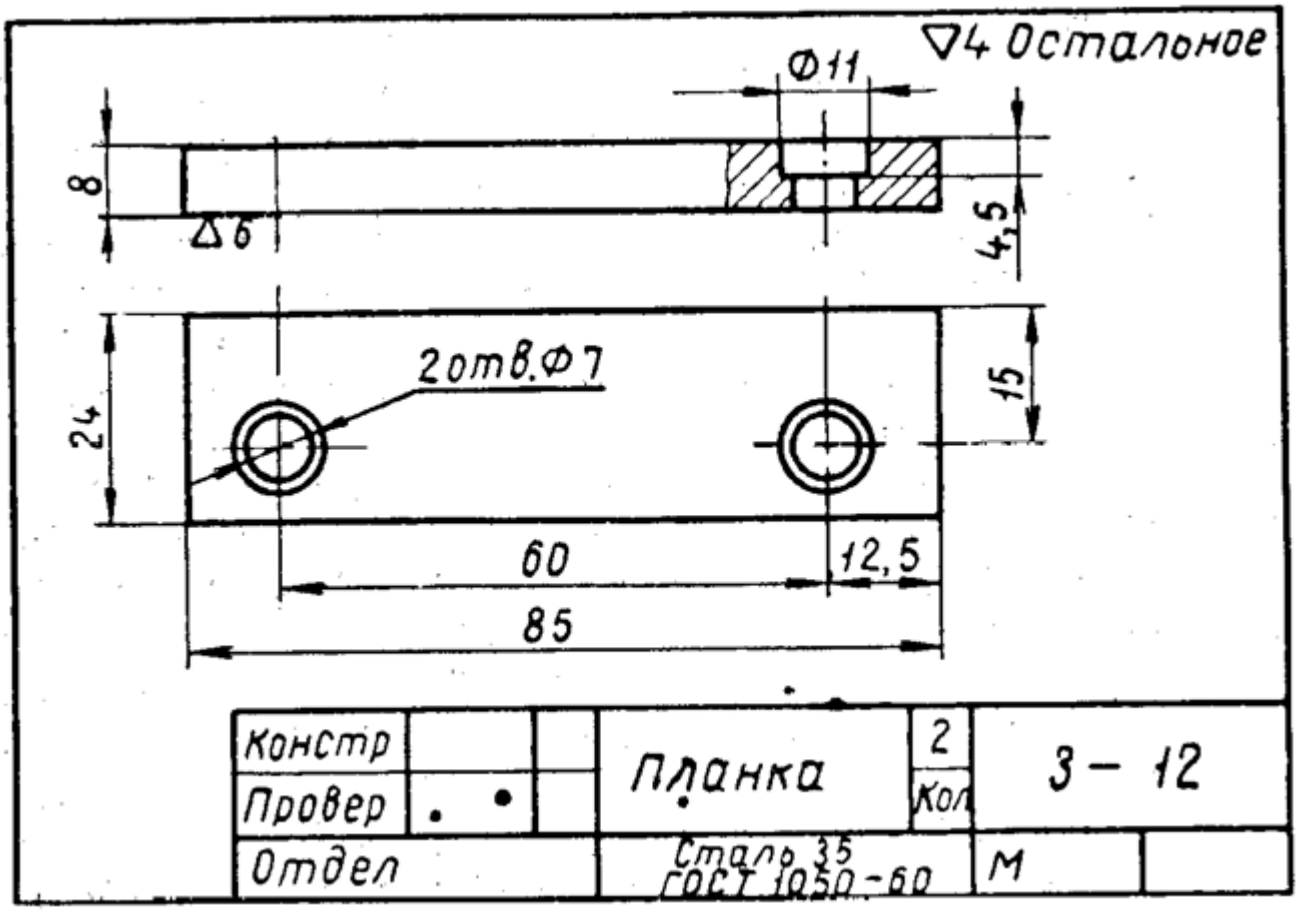
▽5 Остальное



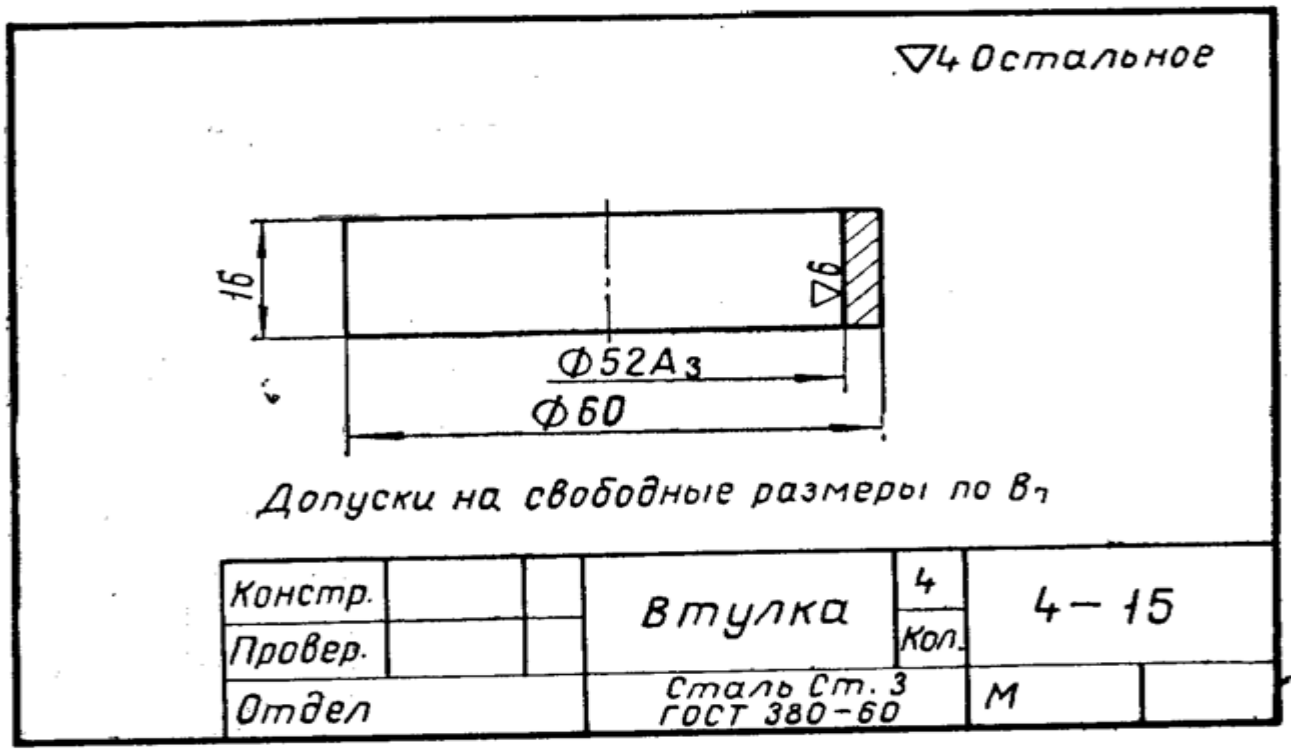

Острые кромки притупить

| | | | | | |
|---|---|---|---|---|---|
| Констр | | | Планка | 1 | 3 – 3 |
| Провер | | | | Кол. | |
| Отдел | | | Сталь 35 ГОСТ 1050-60 | М | |

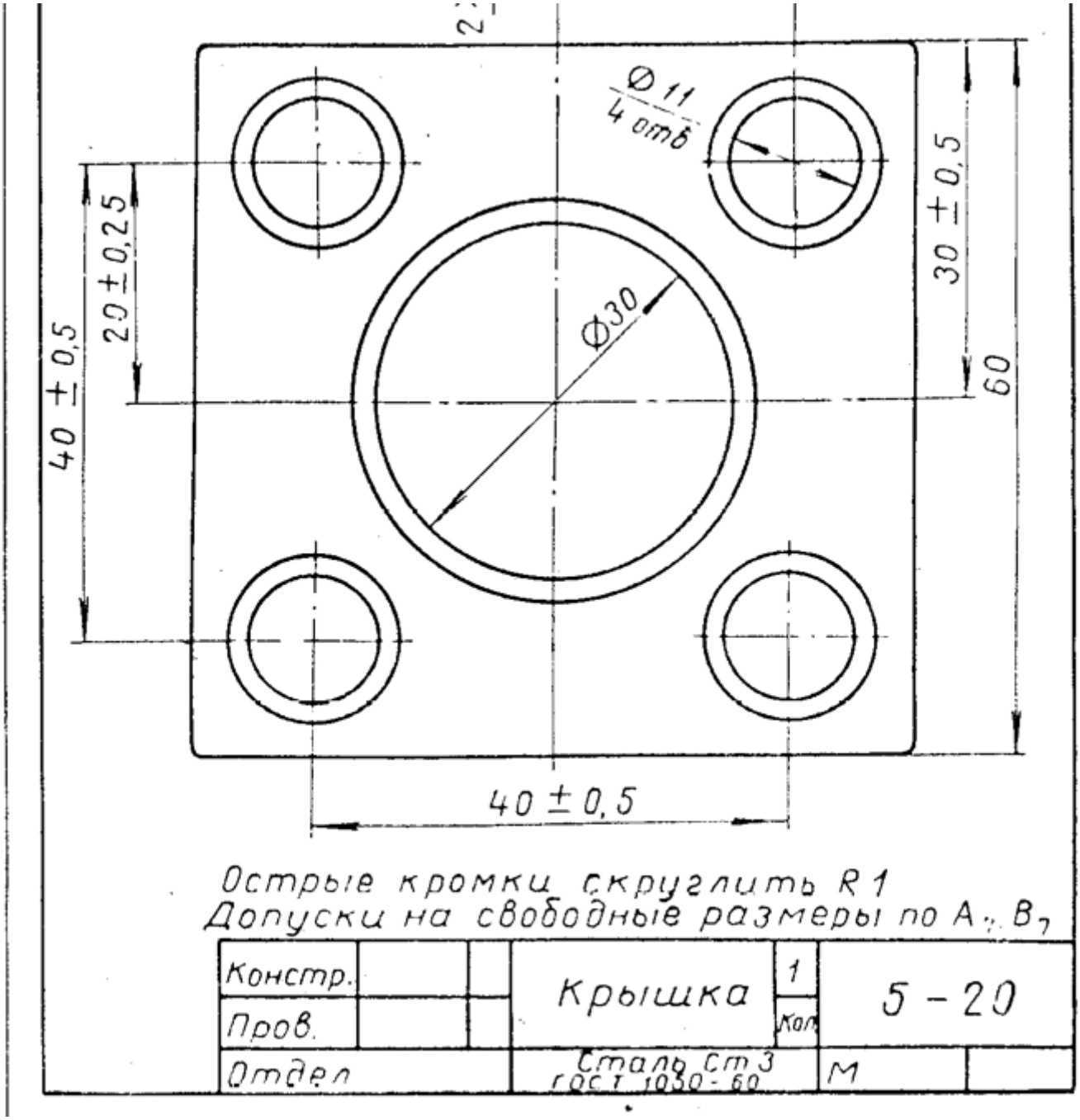
**Вариант 6**.



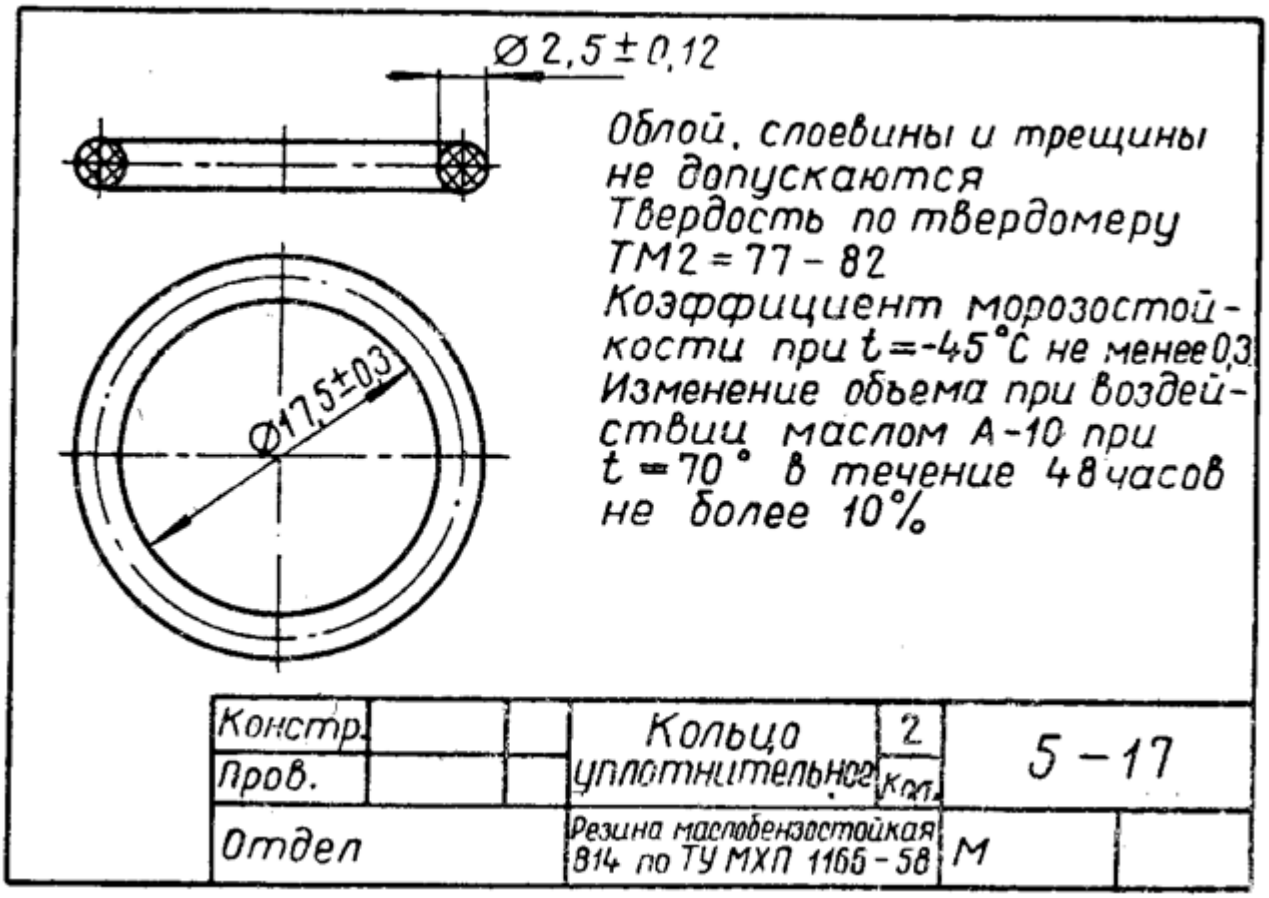

**Вариант 7.**




**Вариант 8.**

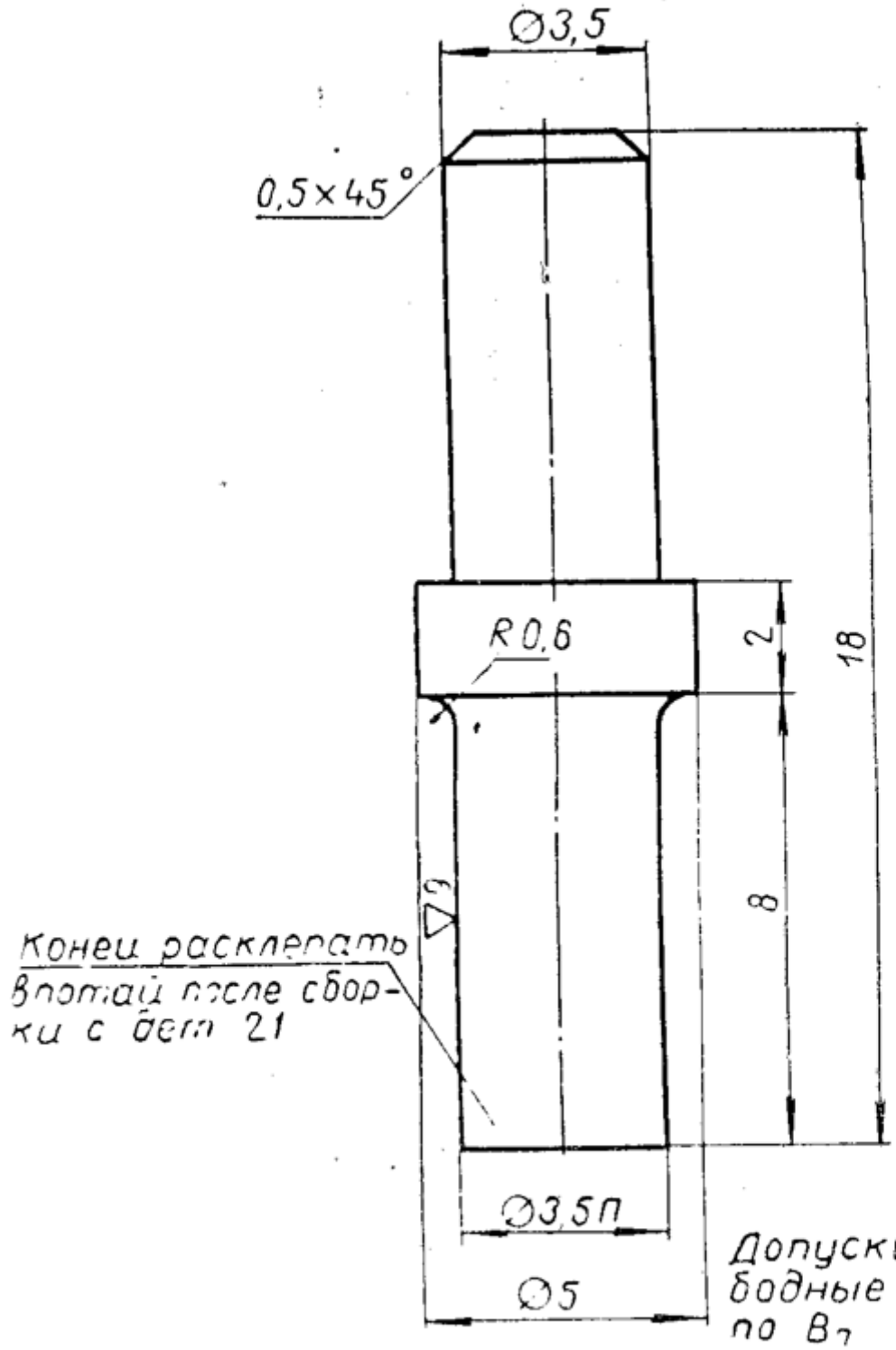

Ø 11
4 отв
30 ± 0,5
60
20 ± 0,25
40 ± 0,5
Ø30
40 ± 0,5
Острые кромки скруглить R 1
Допуски на свободные размеры по A, B,
Констр.
Пров.
Отдел
Крышка
1
Кол
5 - 20
Сталь Ст 3
ГОСТ 1050-60
М

**Вариант 9.**


Ø 2,5 ± 0,12
Ø17,5±0,3
Облой, слоевины и трещины не допускаются
Твердость по твердомеру ТМ2 = 77 – 82
Коэффициент морозостойкости при t = -45°С не менее 0,3
Изменение объема при воздействии маслом А-10 при t = 70° в течение 48 часов не более 10%
Констр.
Пров.
Отдел
Кольцо уплотнительное
2
Кол.
5 – 17
Резина маслобензостойкая 814 по ТУ МХП 1165 – 58
М

**Вариант 10.**

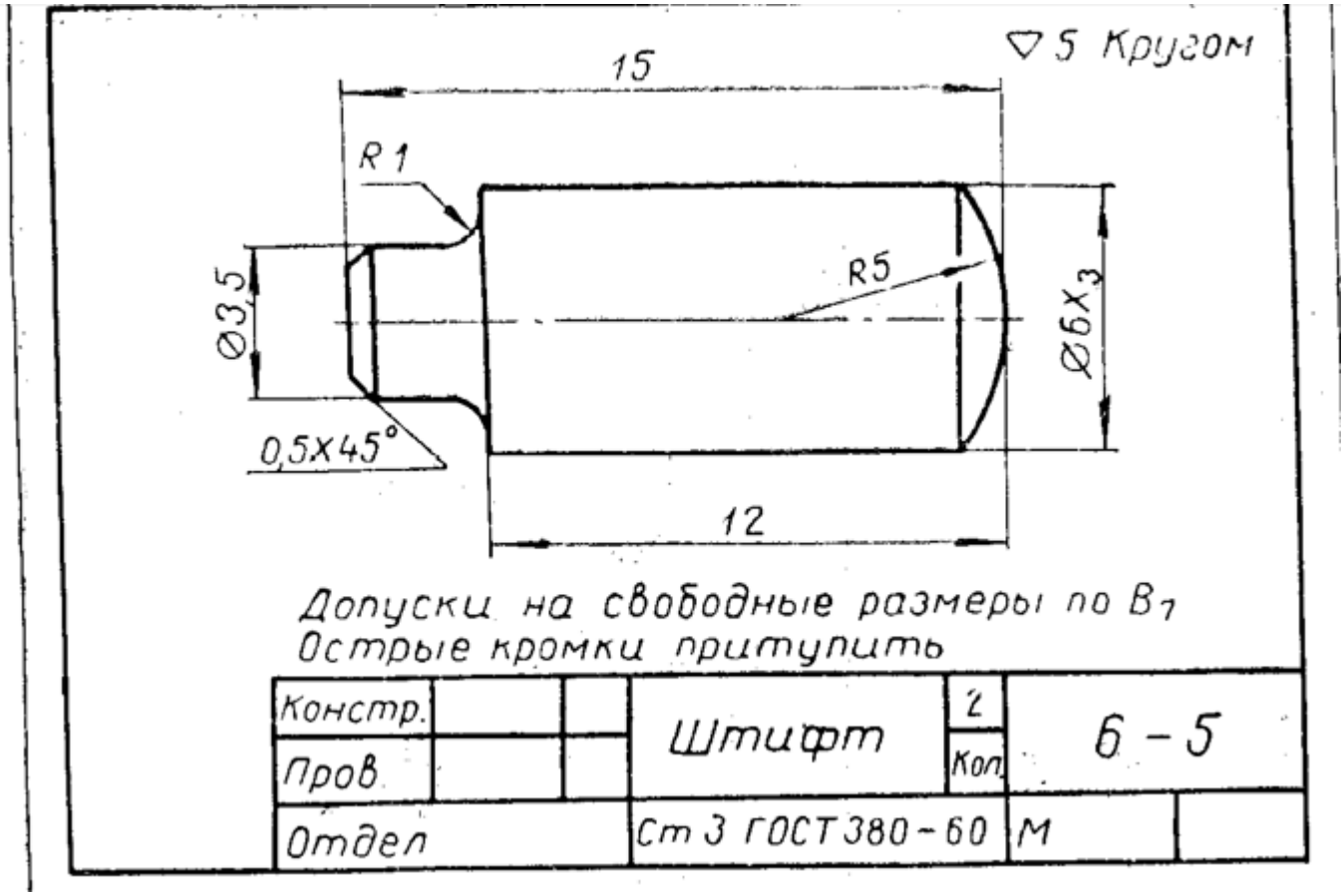
▽3 Остальное



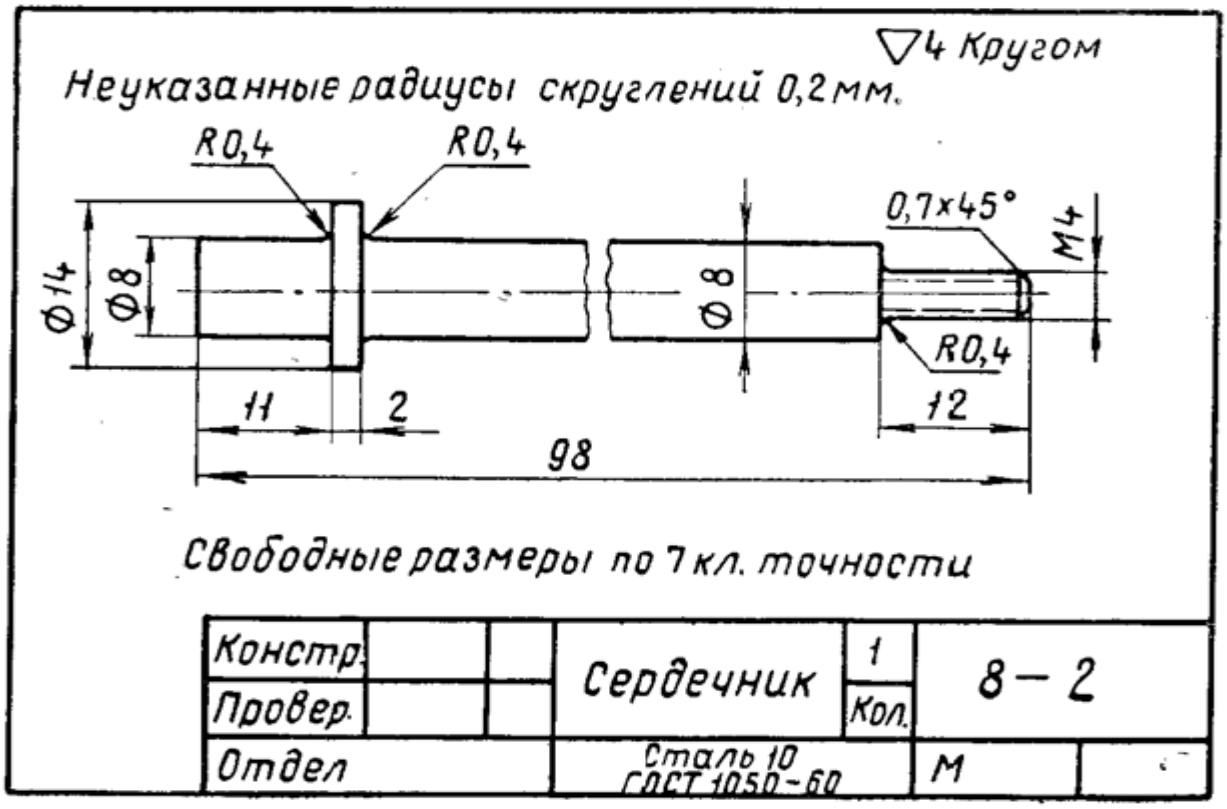

Допуски на свободные размеры по $B_7$

| | | | | | |
|---|---|---|---|---|---|
| Констр. | | | Палец | 2 | 6 – 3 |
| Пров. | | | | Кол. | |
| Отдел | | | Ст3 ГОСТ 380-60 | М | |

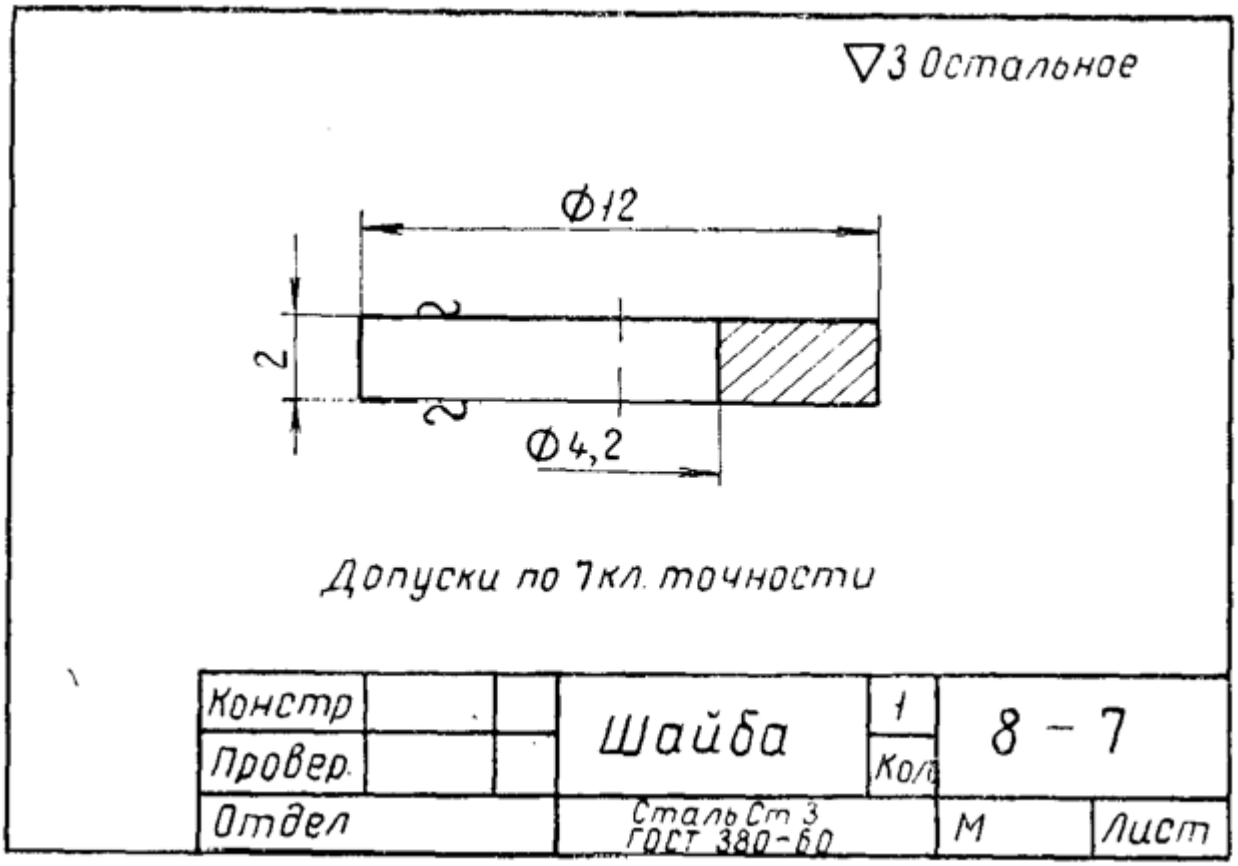
**Вариант 11.**



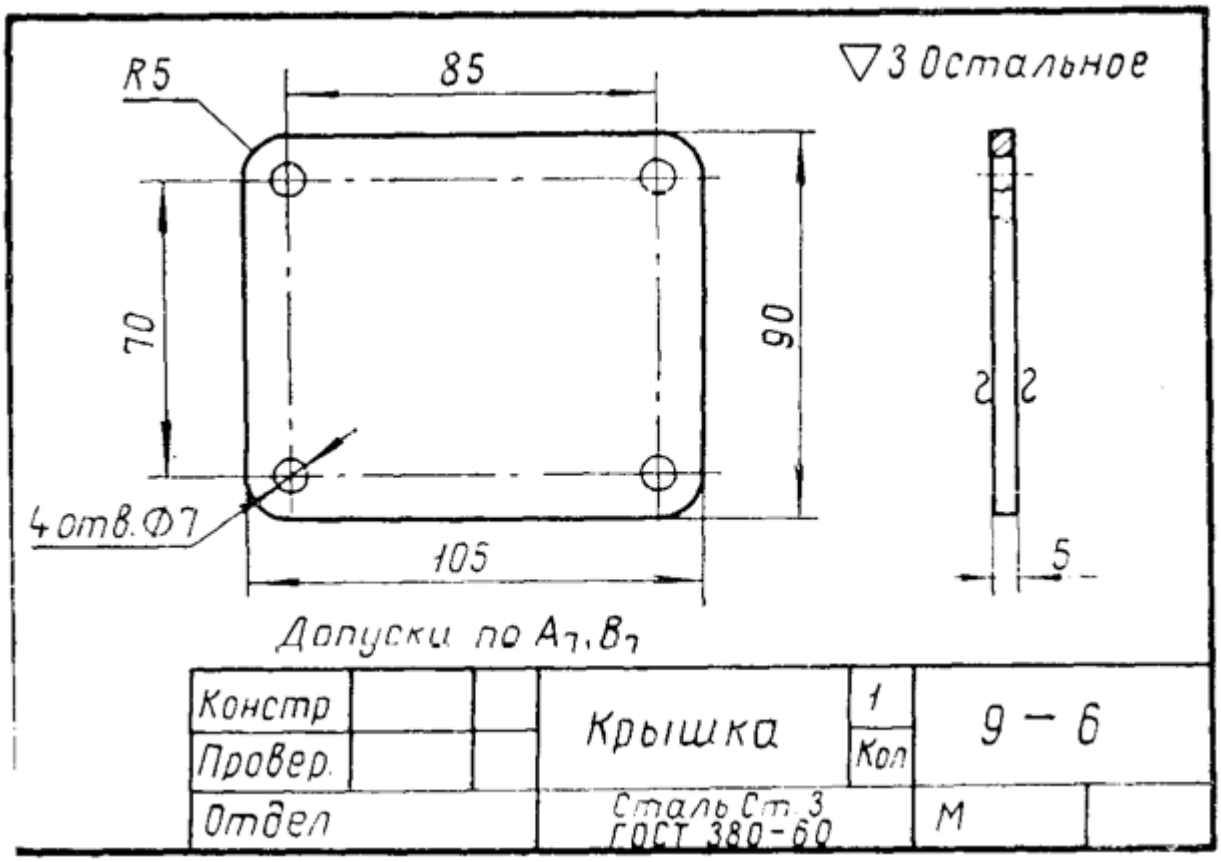

**Вариант 12.**



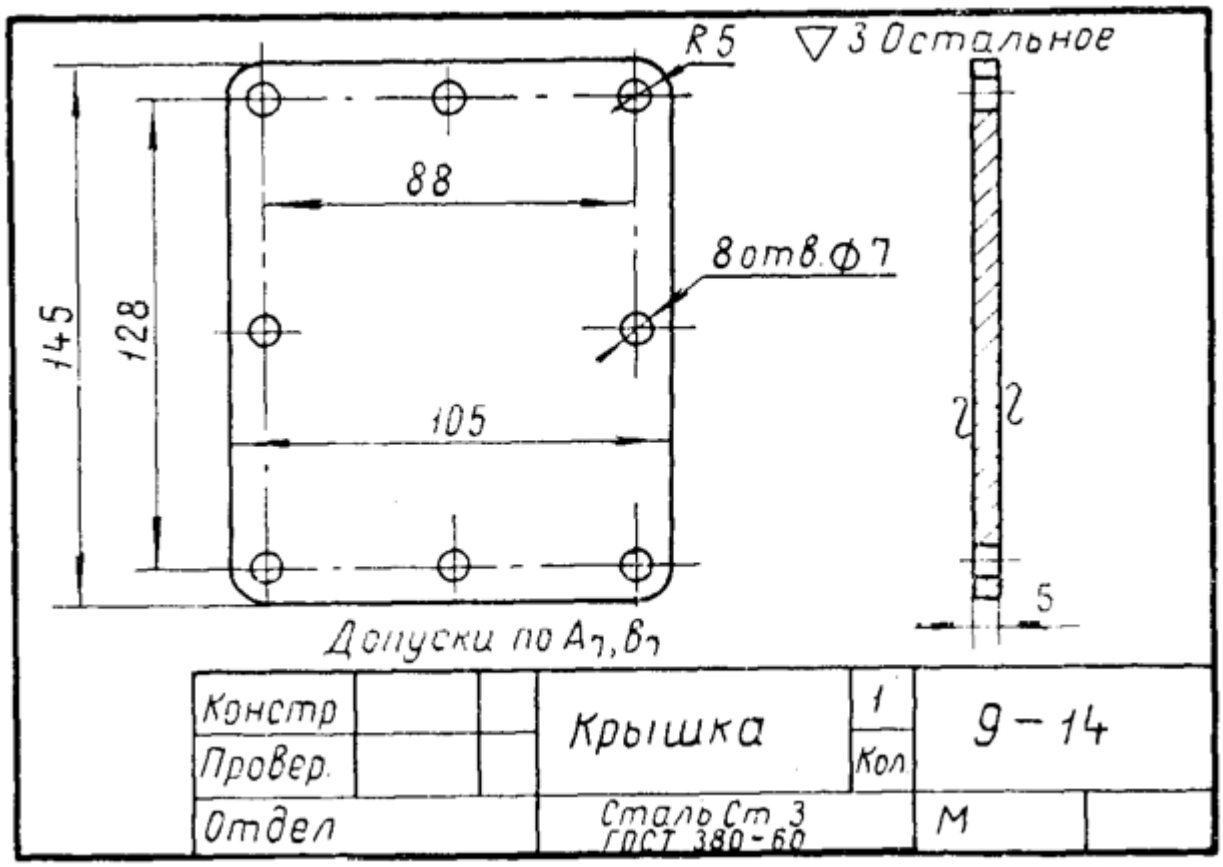

**Вариант 13.**



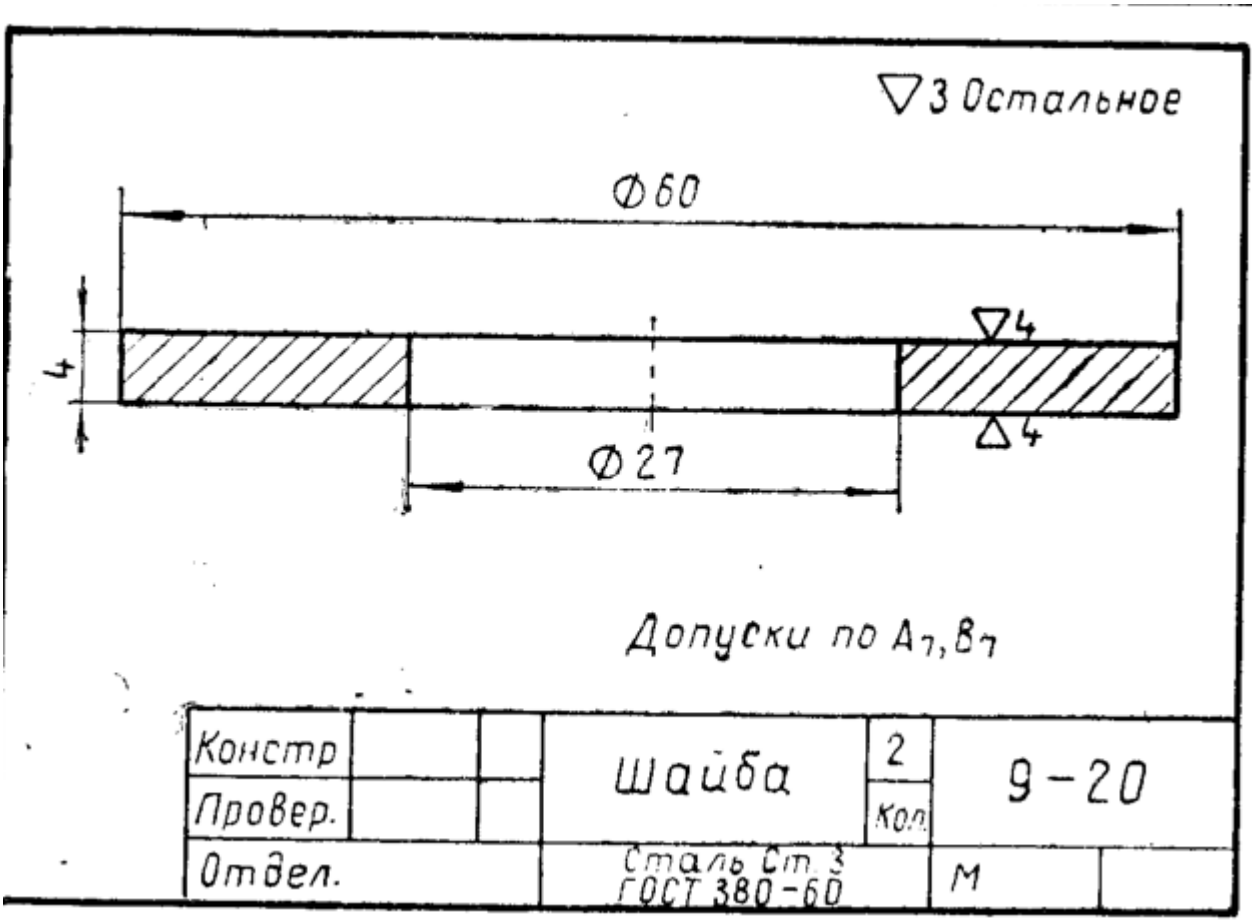

**Вариант 14**



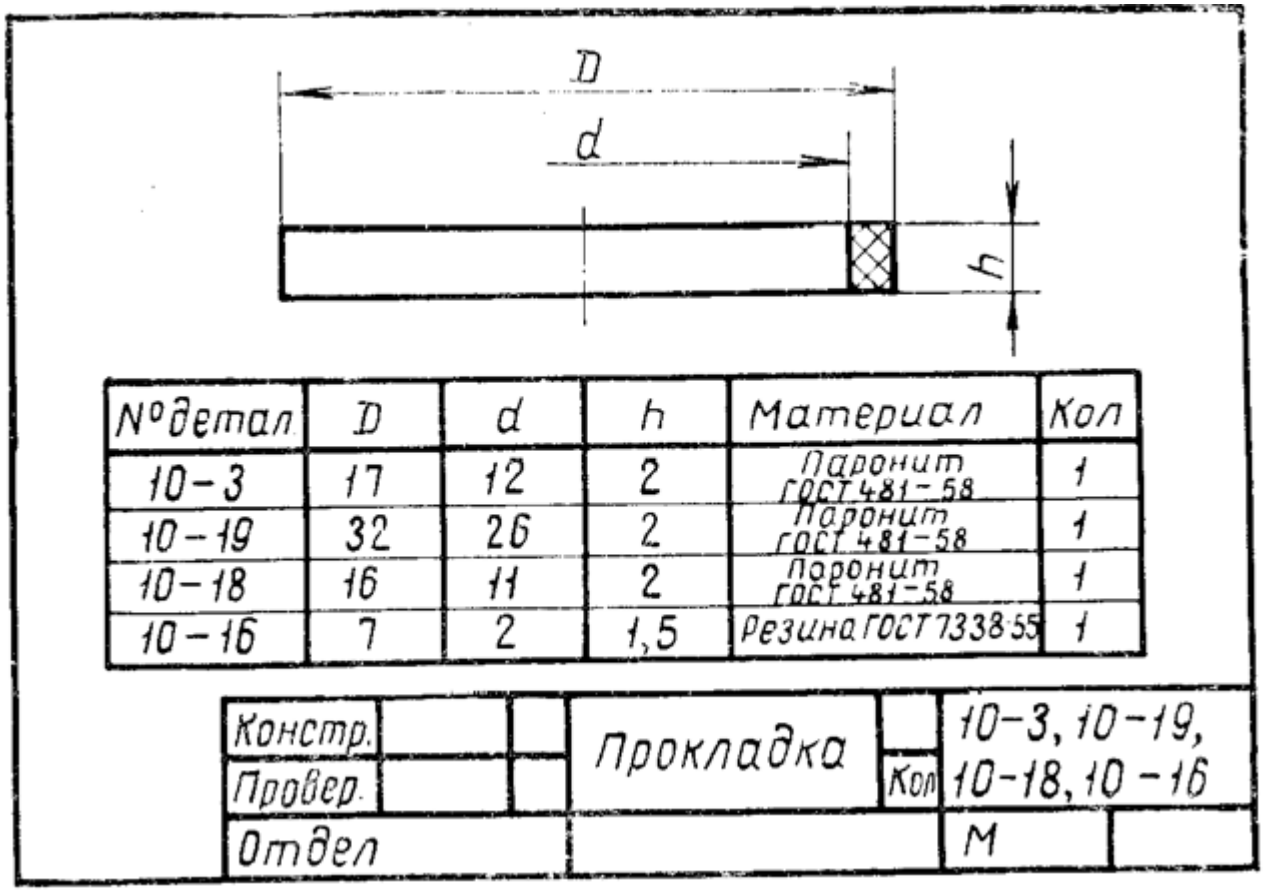

**Вариант 15**




**Вариант 16**



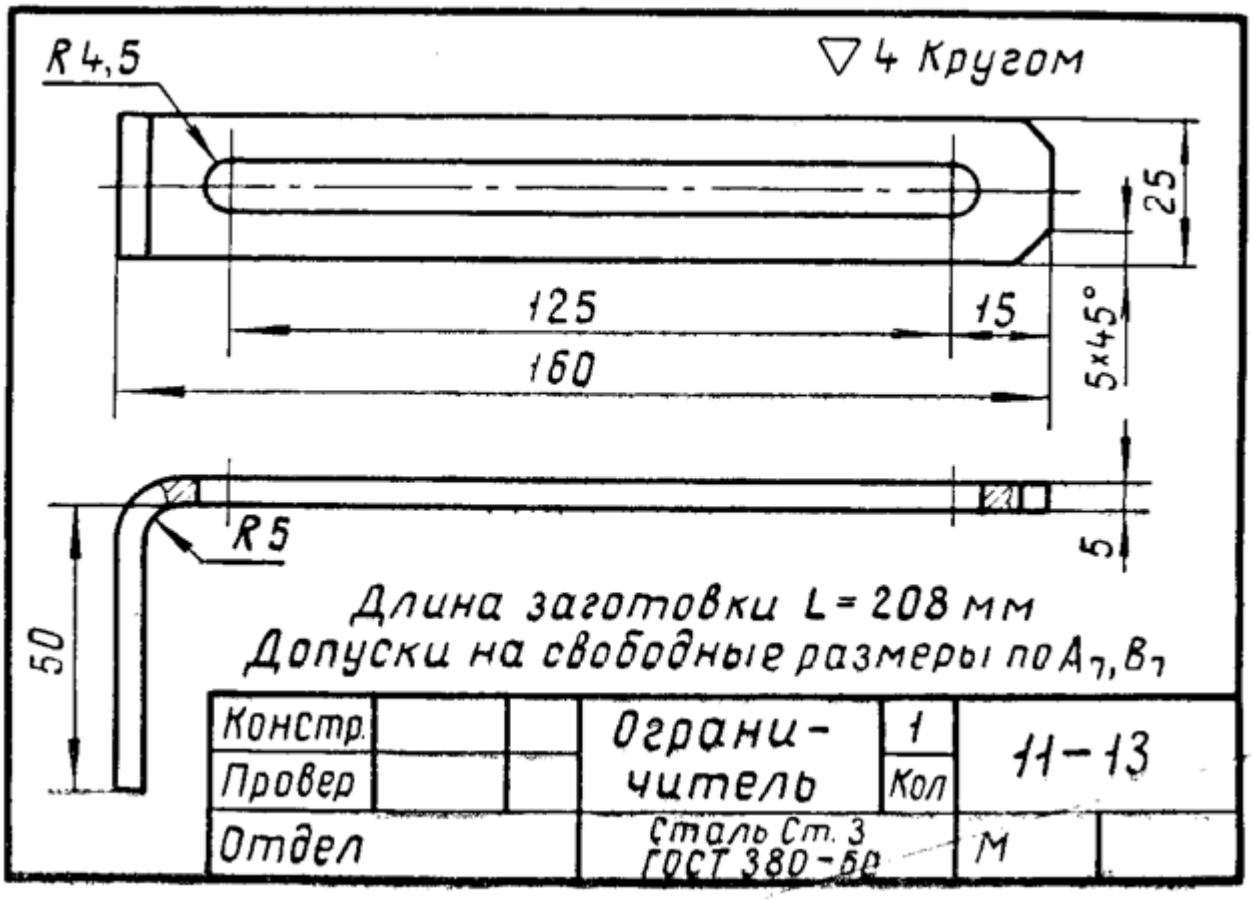

**Вариант 17**




| №детал | D | d | h | Материал | Кол |
|---|---|---|---|---|---|
| 10-3 | 17 | 12 | 2 | Паронит ГОСТ 481-58 | 1 |
| 10-19 | 32 | 26 | 2 | Паронит ГОСТ 481-58 | 1 |
| 10-18 | 16 | 11 | 2 | Паронит ГОСТ 481-58 | 1 |
| 10-16 | 7 | 2 | 1,5 | Резина ГОСТ 7338-55 | 1 |

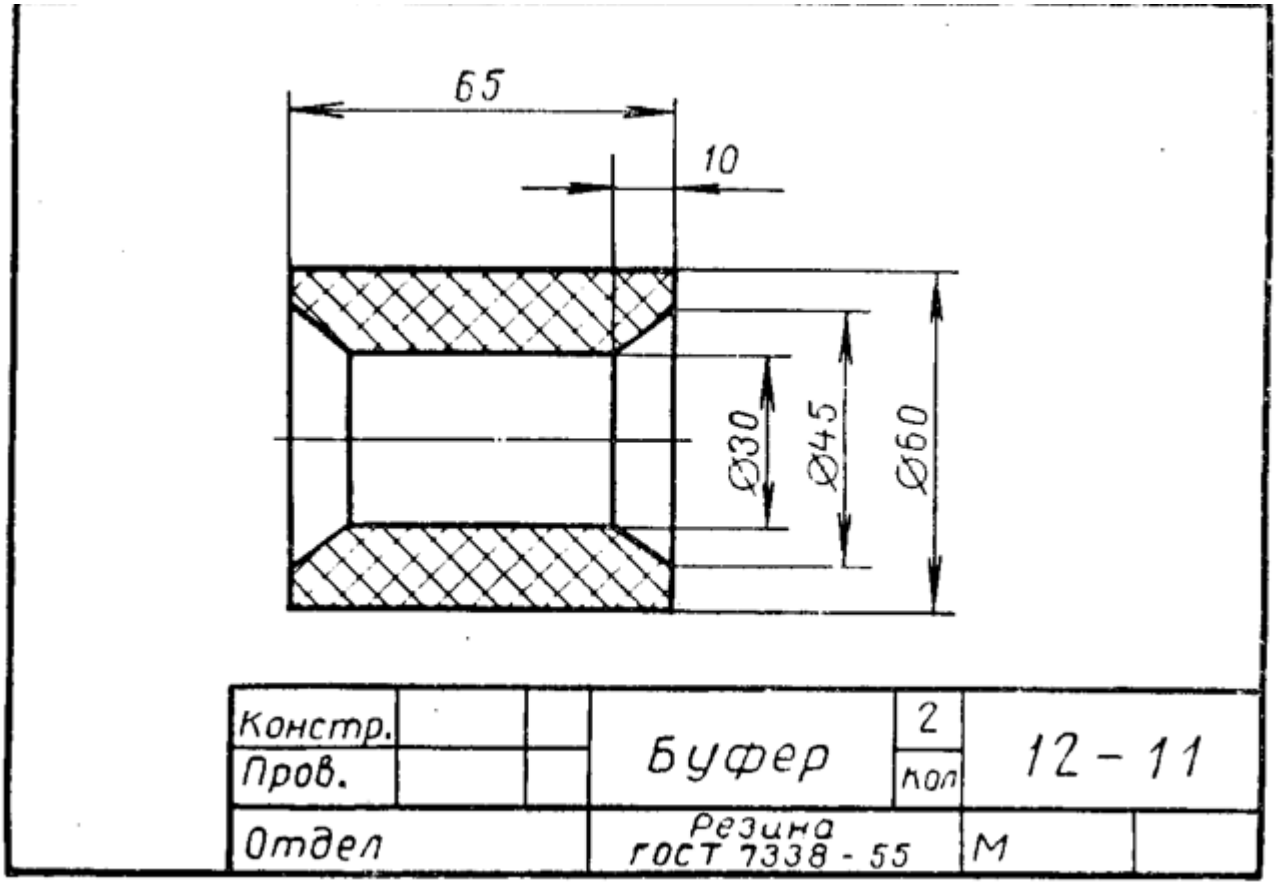
| | | | | | |
|---|---|---|---|---|---|
| Констр. | | | Прокладка | | 10-3, 10-19, |
| Провер. | | | | Кол | 10-18, 10-16 |
| Отдел | | | | М | |

**Вариант 18**

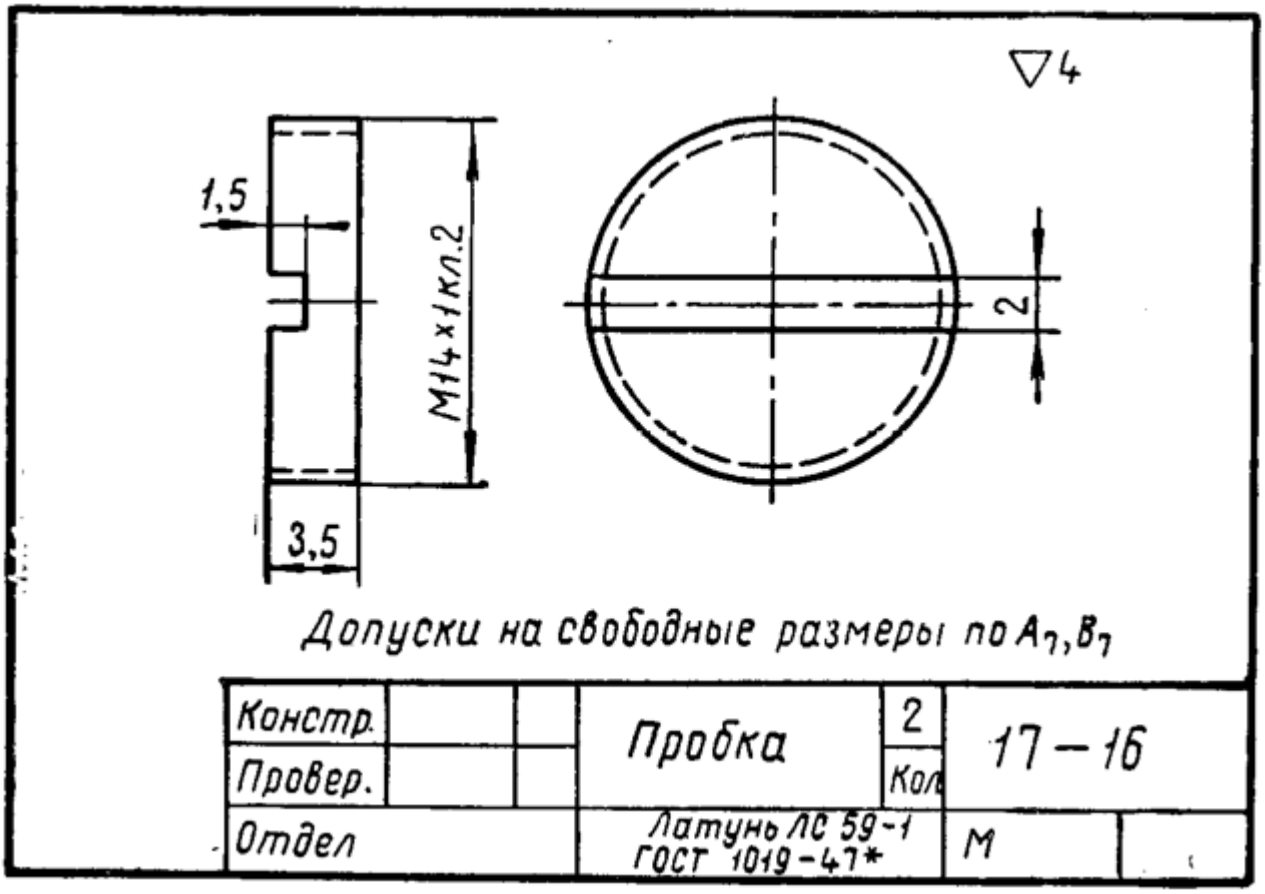



| | | | | | |
|---|---|---|---|---|---|
| Констр. | | | Ограни- | 1 | 11-13 |
| Провер | | | читель | Кол | |
| Отдел | | | Сталь Ст. 3 ГОСТ 380-50 | М | |

**Вариант 19**


65
10
Ø30
Ø45
Ø60
Констр.
Пров.
Отдел
Буфер
2
кол
12 – 11
Резина
ГОСТ 7338 - 55
М

**Вариант 20**


▽4
1,5
М14х1кл.2
3,5
2
Допуски на свободные размеры по $A_7, B_7$
Констр.
Провер.
Отдел
Пробка
2
Кол
17 – 16
Латунь ЛС 59-1
ГОСТ 1019-47*
М