 Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

                        Факультет      Информационных технологий и управления

                        Кафедра         Интеллектуальных информационных технологий

**ОТЧЁТ**

по дисциплине “Графический интерфейс интеллектуальных систем”

Лабораторная работа №2

Выполнил:

Кончатов С.Н., гр. 221703

Проверил:

Сальников Д.А.

Минск 2024

Задание: разработать элементарный графический редактор, реализующий построение линий второго порядка: окружность, эллипс, гипербола, парабола. Выбор кривой задается из пункта меню и доступен через панель инструментов «Линии второго порядка». В редакторе кроме режима генерации линий второго порядка в пользовательском окне должен быть предусмотрен отладочный режим, где отображается пошаговое решение на дискретной сетке.

Ход работы

Для создания программы использовалась библиотека python PyGame.

Код реализации построения окружности

class Brz\_circle\_alg(Line\_Alg):

    # Режим рисования

    def draw(self, draw\_click):

        return super().draw(draw\_click)

    def draw\_line(self,points,check):

        # Получение начальных точек и приращений

        b\_crd,e\_crd=points

        dx=e\_crd[0]-b\_crd[0]

        dy=e\_crd[1]-b\_crd[1]

        lim=b\_crd[1]

        # Начальная ошибка

        r=int((dx\*\*2+dy\*\*2)\*\*0.5)

        delta=2-2\*r

        # Координаты начальной точки

        x=b\_crd[0]

        y=b\_crd[1]+r

        # Отображение начальных точек

        pg.draw.rect(display,BLACK,(x\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

        pg.draw.rect(display,BLACK,(x\*PIXEL,(b\_crd[1]-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

        # Двигаемя по диагонали

        def deqz(x,y,delta):return x+1,y-1,delta+2\*(x-b\_crd[0]+1)-2\*(y-b\_crd[1]-1)+2

        # Отображение

        while y>lim:

            if delta<0:

                d=2\*delta+2\*(y-b\_crd[1])-1

                if d<=0:

                    # Двигаемся по горизонтали

                    x+=1

                    delta=delta+2\*(x-b\_crd[0])+1

                else:x,y,delta=deqz(x,y,delta)

            elif delta>0:

                d=2\*delta-2\*(x-b\_crd[0])-1

                if d>0:

                    # Двигаемся по вертикали

                    y-=1

                    delta=delta-2\*(y-b\_crd[1])+1

                else:x,y,delta=deqz(x,y,delta)

            else:x,y,delta=deqz(x,y,delta)

            # Отображение найденных точек

            pg.draw.rect(display,BLACK,(x\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

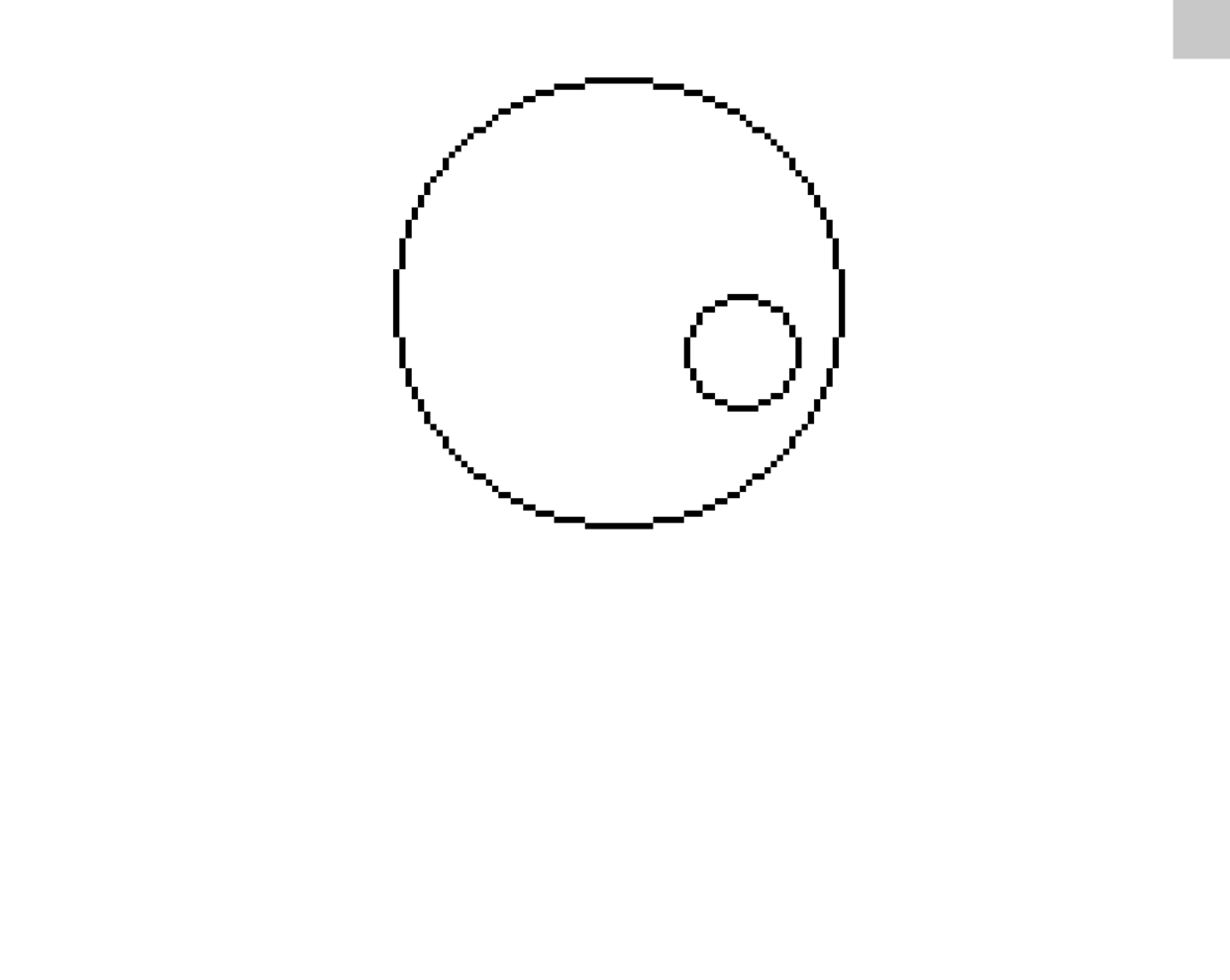
            pg.draw.rect(display,BLACK,(x\*PIXEL,(b\_crd[1]-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

            pg.draw.rect(display,BLACK,((b\_crd[0]-x+b\_crd[0])\*PIXEL,(b\_crd[1]-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

            pg.draw.rect(display,BLACK,((b\_crd[0]-x+b\_crd[0])\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

            # Отладка

            check()



Код реализации построения эллипса

class Ellipse\_alg(Line\_Alg):

    # Режим рисования

    def draw(self, draw\_click):

        return super().draw(draw\_click)

    def draw\_line(self,points,check):

        # Получение начальных точек и приращения

        b\_crd,e\_crd=points

        a=e\_crd[0]-b\_crd[0]

        b=e\_crd[1]-b\_crd[1]

        lim=b\_crd[1]

        # Начальная ошибка и координаты начальной точки

        delta=a\*\*2\*(1-2\*abs(b))+b\*\*2

        x=b\_crd[0]

        y=b\_crd[1]+abs(b)

        # Отображение начальных точек

        pg.draw.rect(display,BLACK,(x\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

        pg.draw.rect(display,BLACK,(x\*PIXEL,(b\_crd[1]-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

        # Двигаемя по диагонали

        def deqz(x,y,delta):return x+1,y-1,delta+(2\*(x-b\_crd[0])+1)\*b\*\*2+(-2\*(y-b\_crd[1])+1)\*a\*\*2

        # Отображение

        while y>lim:

            if delta<0:

                d=2\*delta+2\*(y-b\_crd[1])\*a\*\*2-1

                if d<=0:

                    # Двигаемя по горизонтали

                    x+=1

                    delta=delta+b\*\*2\*(2\*(x-b\_crd[0])+1)

                else:x,y,delta=deqz(x,y,delta)

            elif delta>0:

                d=2\*delta-2\*(x-b\_crd[0])\*b\*\*2-1

                if d>0:

                    # Двигаемя по вертикали

                    y-=1

                    delta=delta+a\*\*2\*(-2\*(y-b\_crd[1])+1)

                else:x,y,delta=deqz(x,y,delta)

            else:x,y,delta=deqz(x,y,delta)

            # Отображение полученных точек

            pg.draw.rect(display,BLACK,(x\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

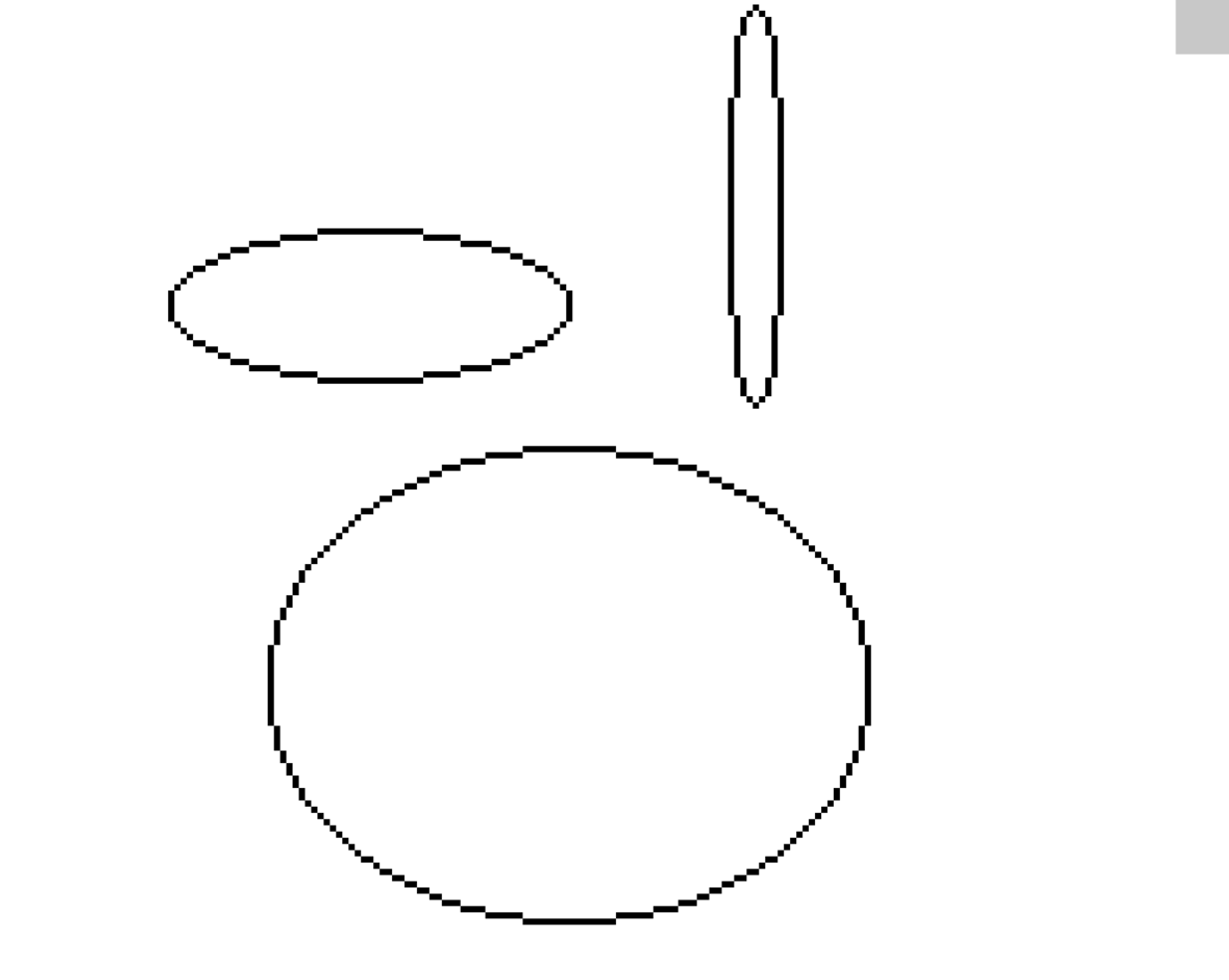
            pg.draw.rect(display,BLACK,(x\*PIXEL,(b\_crd[1]-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

            pg.draw.rect(display,BLACK,((b\_crd[0]-x+b\_crd[0])\*PIXEL,(b\_crd[1]-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

            pg.draw.rect(display,BLACK,((b\_crd[0]-x+b\_crd[0])\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

            # Отладка

            check()



Код реализации построения гиперболы

class Giperb\_alg(Line\_Alg):

    # Режим рисования

    def draw(self, draw\_click):

        return super().draw(draw\_click)

    def draw\_line(self,points,check):

        # Получение заданных точек и значения а

        b\_crd,e\_crd=points

        a=int((e\_crd[0]-b\_crd[0])/2)

        # Проверка валидности

        if a != 0:

            # Получение значения b

            b=abs(e\_crd[1]-b\_crd[1])

            # Начальная ошибка и координаты начальной точки

            delta=b\*\*2\*(1+2\*abs(a))-a\*\*2

            x,y=abs(a),0

            # Отображение начальных точек

            pg.draw.rect(display,BLACK,((x+b\_crd[0]+a)\*PIXEL,(y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

            pg.draw.rect(display,BLACK,((-x+b\_crd[0]+a)\*PIXEL,(y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

            # Двигаемся по диагонали

            def deqz(x,y,delta):return x+1,y+1,delta+(2\*x+1)\*b\*\*2-(2\*y+1)\*a\*\*2

            # Выбор ограничения

            if abs(a)<=b:cond=lambda:y<b

            else:cond=lambda:x<a+10

            # Отображение

            while cond:

                if delta<0:

                    d=2\*delta+2\*y\*a\*\*2+1

                    if d<=0:

                        # Двигаемся по горизонтали

                        x+=1

                        delta=delta+b\*\*2\*(2\*x+1)

                    else:x,y,delta=deqz(x,y,delta)

                elif delta>0:

                    d=2\*delta-2\*x\*b\*\*2-1

                    if d>0:

                        # Двигаемся по вертикали

                        y+=1

                        delta=delta-a\*\*2\*(2\*y+1)

                    else:x,y,delta=deqz(x,y,delta)

                else:x,y,delta=deqz(x,y,delta)

                # Отображение полученных точек

                pg.draw.rect(display,BLACK,((x+b\_crd[0]+a)\*PIXEL,(y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

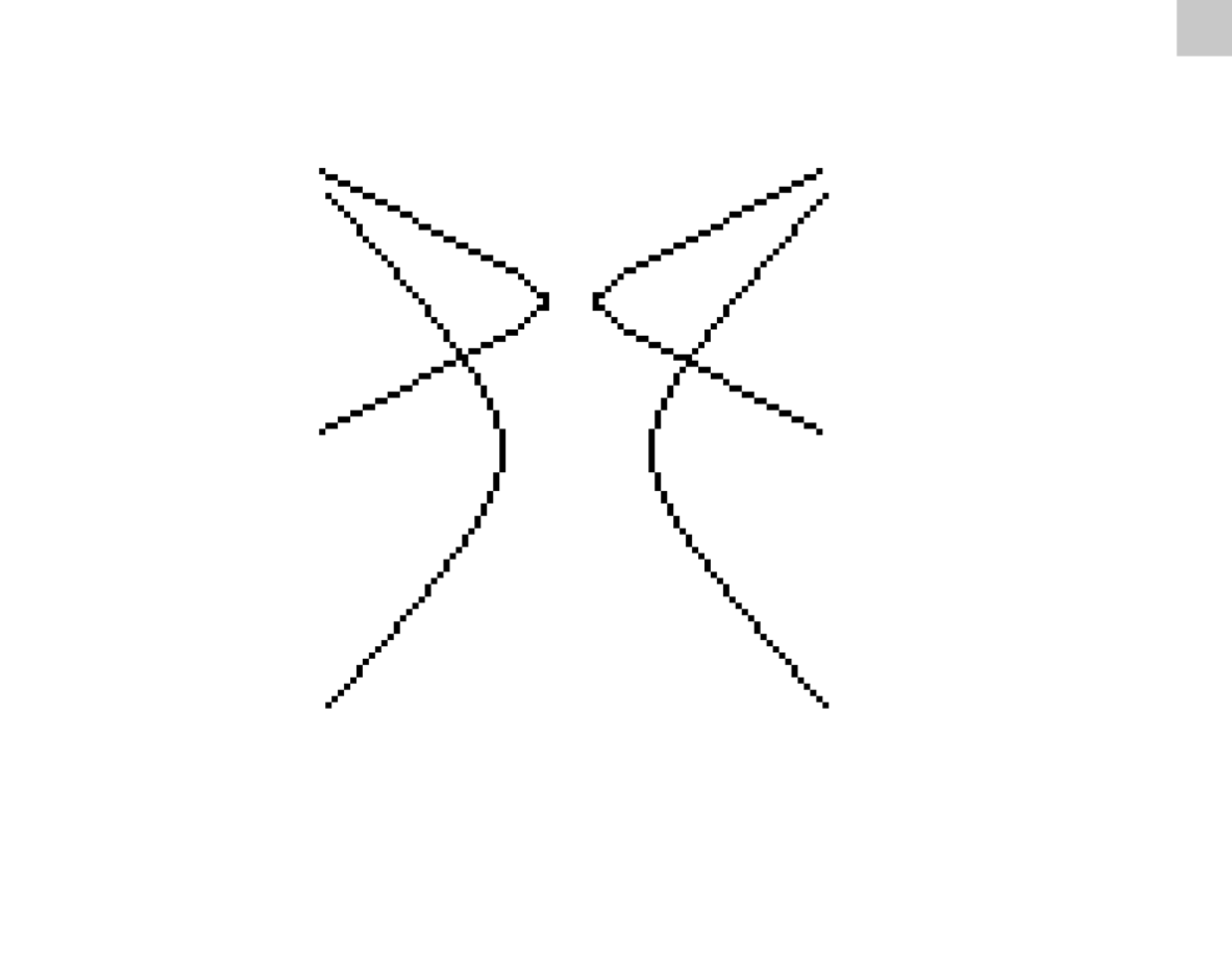
                pg.draw.rect(display,BLACK,((x+b\_crd[0]+a)\*PIXEL,(-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

                pg.draw.rect(display,BLACK,((-x+b\_crd[0]+a)\*PIXEL,(-y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

                pg.draw.rect(display,BLACK,((-x+b\_crd[0]+a)\*PIXEL,(y+b\_crd[1])\*PIXEL,PIXEL,PIXEL))

                # Отладка

                check()



Код построения параболы

class Parab\_alg(Line\_Alg):

    # Режим рисования

    def draw(self, draw\_click):

        return super().draw(draw\_click)

    def draw\_line(self,points,check):

        # Получение заданных точек и значения b

        b\_crd,e\_crd=points

        b=e\_crd[1]-b\_crd[1]

        # Проверка валидности

        if b!=0:

            # Получение знака построения

            if b<0:ty=-1

            else:ty=1

            # Значение а, предела, начальной ошибки, координаты начальной точки

            a=e\_crd[0]-b\_crd[0]

            lim=e\_crd[1]

            delta=2\*abs(b)-4\*a\*\*2

            x,y=b\_crd

            # Обображение начальной точки

            pg.draw.rect(display,BLACK,(x\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

            # Двигаемся по диагонали

            def deqz(x,y,delta):return x+1,y+ty,delta-4\*a\*\*2+2\*abs(b)\*(2\*(x-b\_crd[0])+1)

            # Отображение

            while y\*ty<lim\*ty:

                if delta>0:

                    d=2\*delta+4\*a\*\*2

                    if d>0:

                        # Двигаемся по вертикали

                        y+=ty

                        delta=delta-4\*a\*\*2

                    else:x,y,delta=deqz(x,y,delta)

                elif delta<0:

                    d=2\*delta-2\*abs(b)\*(2\*(x-b\_crd[0])+1)

                    if d<=0:

                        # Двигаемся по горизонтали

                        x+=1

                        delta=delta+2\*abs(b)\*(2\*(x-b\_crd[0])+1)

                    else:x,y,delta=deqz(x,y,delta)

                else:x,y,delta=deqz(x,y,delta)

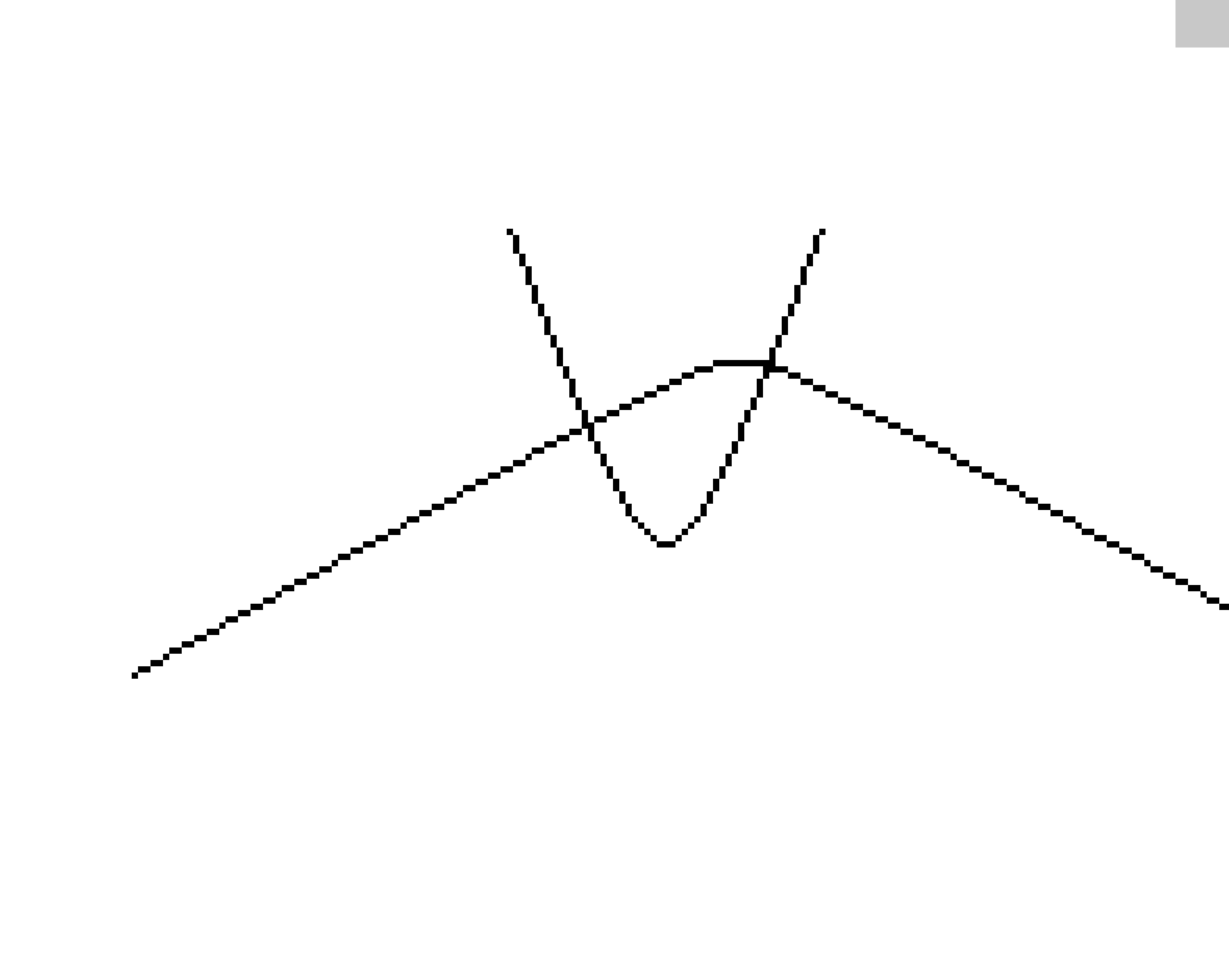
                # Отображение полученных точек

                pg.draw.rect(display,BLACK,(x\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

                pg.draw.rect(display,BLACK,((b\_crd[0]-x+b\_crd[0])\*PIXEL,y\*PIXEL,PIXEL,PIXEL))

                # Отладка

                check()



Вывод

В ходе данной лабораторной работы были получены знания об алгоритмах построении линий второго порядка, данные знания были применены на практике и реализованы в среде Python.