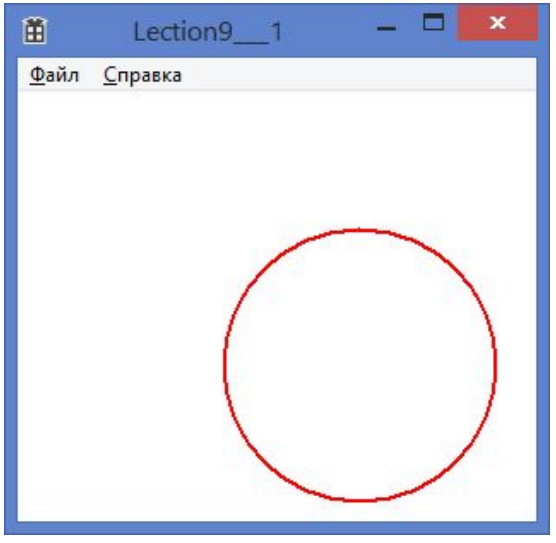
Основы программирования Лабораторная работа №11

WinAPI Графика – Рекурсия

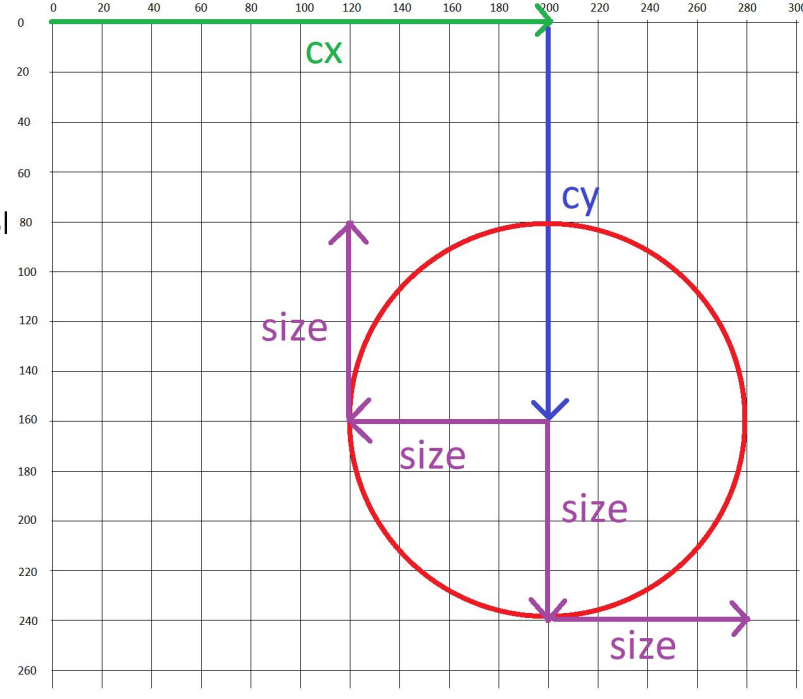
Власенко Олег Федосович

Рисование окружности

```
void Circle(HDC hdc, int cx, int cy, int size) {
        HPEN hPen;
        hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
        SelectObject(hdc, hPen);
        Ellipse(hdc, cx - size, cy - size, cx + size, cy + size);

        DeleteObject(hPen);
}
...
```

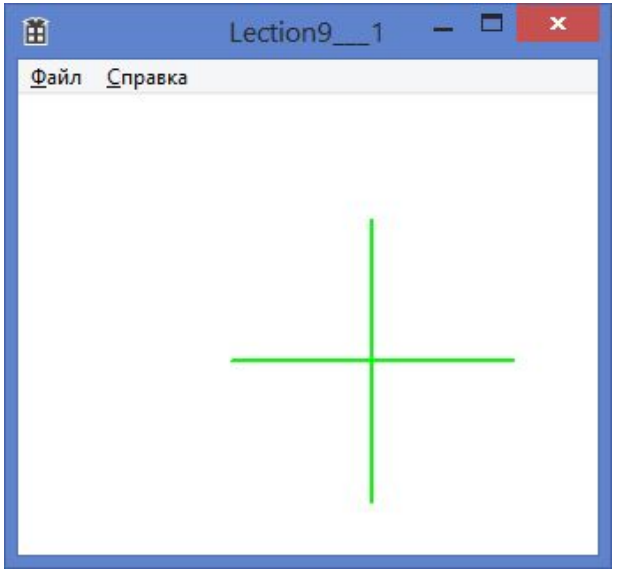


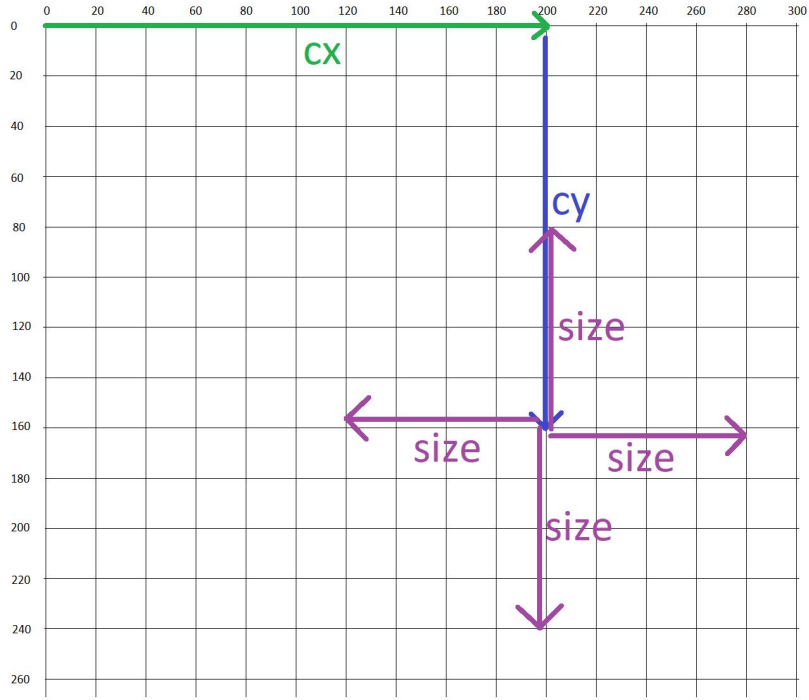

```
 case WM_PAINT:
      {
         PAINTSTRUCT ps;
         HDC hdc = BeginPaint(hWnd, &ps);
         // TODO: Добавьте сюда любой код прорисовк
         // использующий HDC...
         Circle(hdc, 200, 160, 80);
         EndPaint(hWnd, &ps);
      }
      break;
```



Рисование креста

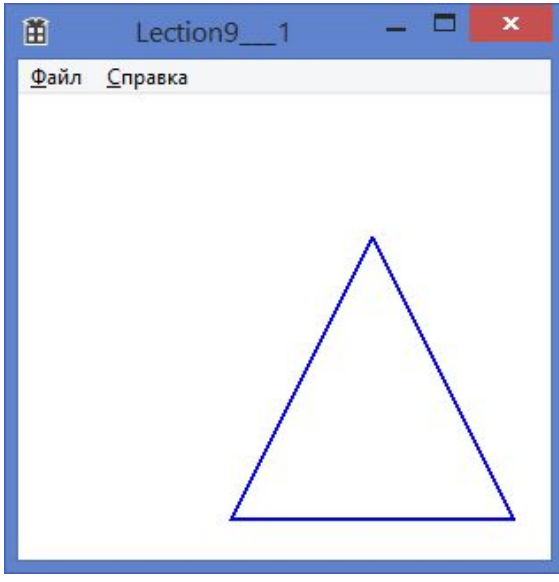
```
void Cross(HDC hdc, int cx, int cy, int size) {
        HPEN hPen;
        hPen = CreatePen(PS_SOLID, 2, RGB(0, 255, 0));
        SelectObject(hdc, hPen);
        MoveToEx(hdc, cx - size, cy, NULL);
        LineTo(hdc, cx + size, cy);
        MoveToEx(hdc, cx, cy - size, NULL);
        LineTo(hdc, cx, cy + size);
        DeleteObject(hPen);
}
 …
 case WM_PAINT:
      {
         PAINTSTRUCT ps;
         HDC hdc = BeginPaint(hWnd, &ps);
         // TODO: Добавьте сюда любой код прорисов
         Cross(hdc, 200, 160, 80);
         EndPaint(hWnd, &ps);
      }
      break;
```

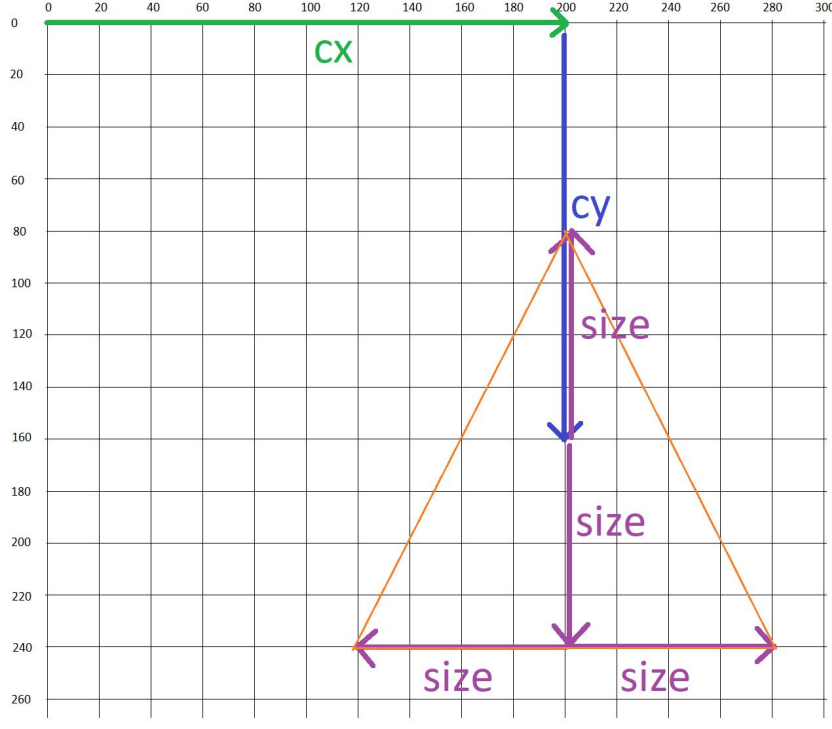




Рисование треугольника

```
void Triangle(HDC hdc, int cx, int cy, int size) {
        HPEN hPen;
        hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
        SelectObject(hdc, hPen);

        MoveToEx(hdc, cx, cy - size, NULL);
        LineTo(hdc, cx + size, cy + size);
        LineTo(hdc, cx - size, cy + size);
        LineTo(hdc, cx, cy - size);

        DeleteObject(hPen);
}
…
 case WM_PAINT:
      {
         PAINTSTRUCT ps;
         HDC hdc = BeginPaint(hWnd, &ps);
         // TODO: Добавьте сюда любой код прорисс
         Triangle(hdc, 200, 160, 80);
         EndPaint(hWnd, &ps);
      }
      break;
```





Рисование треугольника 2

```
void Triangle2(HDC hdc, int cx, int cy, int size) {
        int x1 = cx - size;
        int y1 = cy - size;

        int x2 = cx + size;
        int y2 = cy - size;

        int x3 = cx;
        int y3 = cy + size;

        HPEN hPen;
        hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
        SelectObject(hdc, hPen);
        MoveToEx(hdc, x1, y1, NULL);
        LineTo(hdc, x2, y2);
        LineTo(hdc, x3, y3);
        LineTo(hdc, x1, y1);
        DeleteObject(hPen);
}
…
Triangle2 (hdc, 200, 160, 80);
```

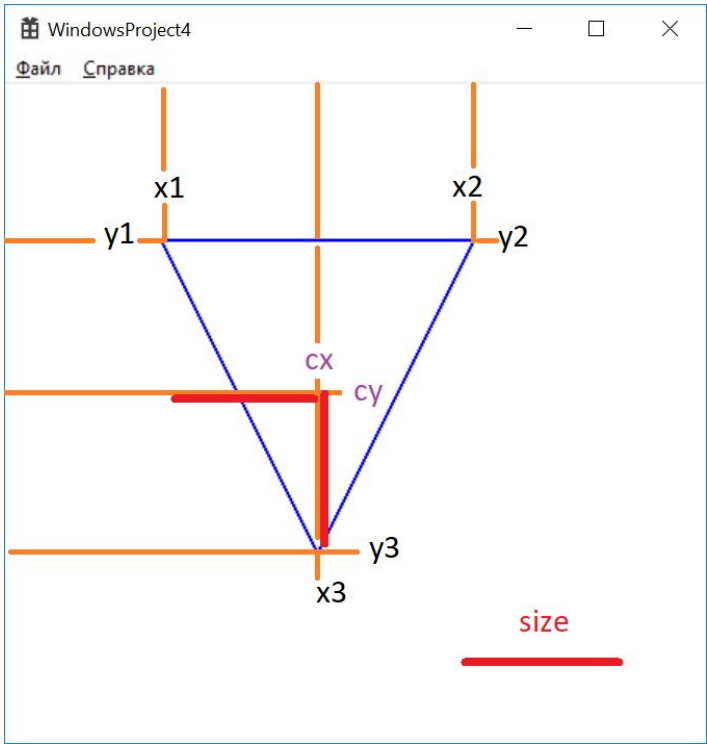
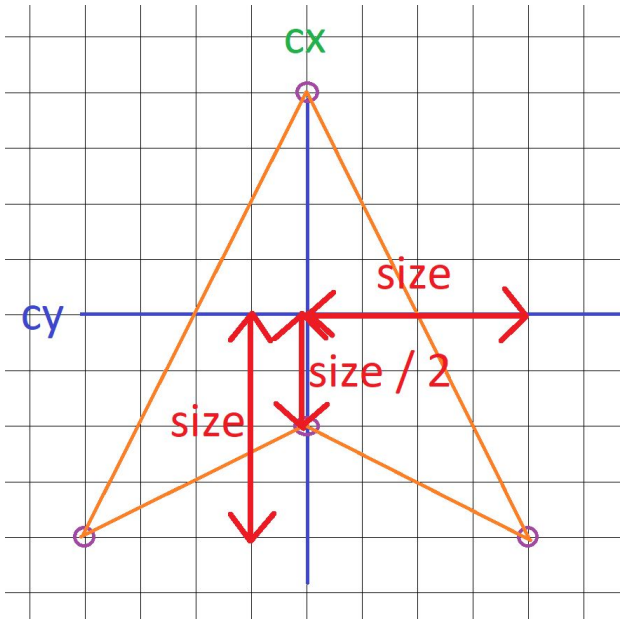


Image1

```
void Image1(HDC hdc, int cx, int cy, int size) {
	int x1 = cx;
	int y1 = cy - size;

	int x2 = cx + size;
	int y2 = cy + size;

	int x3 = cx;
	int y3 = cy + size / 2;

	int x4 = cx - size;
	int y4 = cy + size;

	HPEN hPen;
	hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
	SelectObject(hdc, hPen);

	MoveToEx(hdc, x1, y1, NULL);
	LineTo(hdc, x2, y2);
	LineTo(hdc, x3, y3);
	LineTo(hdc, x4, y4);
	LineTo(hdc, x1, y1);

	DeleteObject(hPen);
}
```



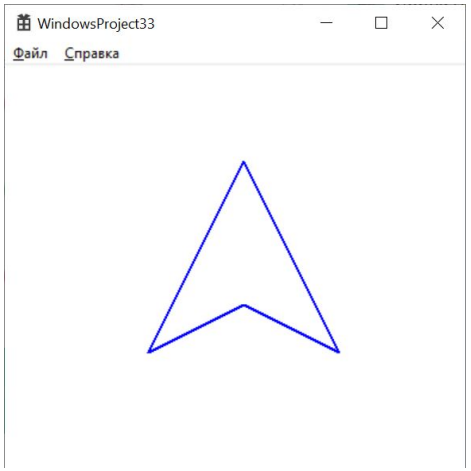
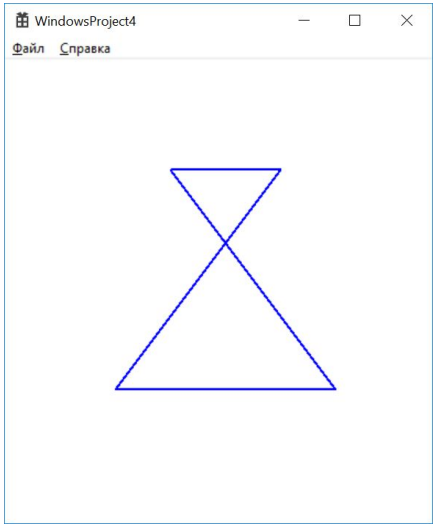


Image2

```
void Image2(HDC hdc, int cx, int cy, int size) {
    int x1 = cx - size / 2;
    int y1 = cy - size;
    int x2 = cx + size / 2;
    int y2 = cy - size;
    int x3 = cx - size;
    int y3 = cy + size;
    int x4 = cx + size;
    int y4 = cy + size;

    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
    SelectObject(hdc, hPen);

    MoveToEx(hdc, x1, y1, NULL);
    LineTo(hdc, x2, y2);
    LineTo(hdc, x3, y3);
    LineTo(hdc, x4, y4);
    LineTo(hdc, x1, y1);

    DeleteObject(hPen);
}
```



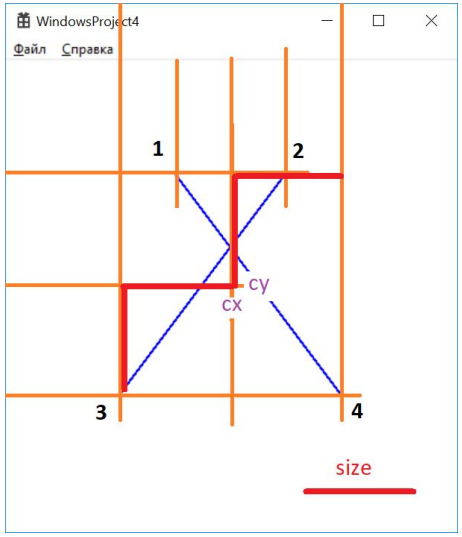


Image3

```
void Image3(HDC hdc, int cx, int cy, int size) {
    int x1 = cx;
    int y1 = cy - size;

    int x2 = cx + size;
    int y2 = cy;

    int x3 = cx;
    int y3 = cy + size;

    int x4 = cx - size;
    int y4 = cy;

    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
    SelectObject(hdc, hPen);

    MoveToEx(hdc, x1, y1, NULL);
    LineTo(hdc, x2, y2);
    LineTo(hdc, x3, y3);
    LineTo(hdc, x4, y4);
    LineTo(hdc, x1, y1);

    DeleteObject(hPen);
}
```

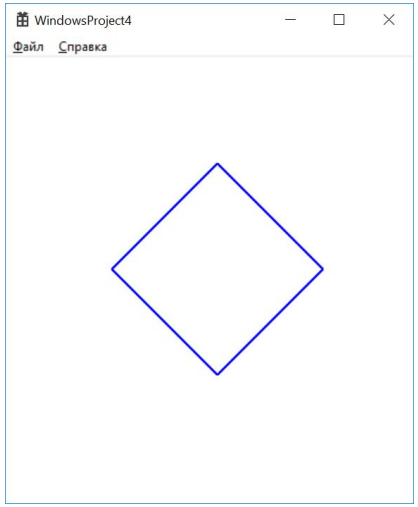
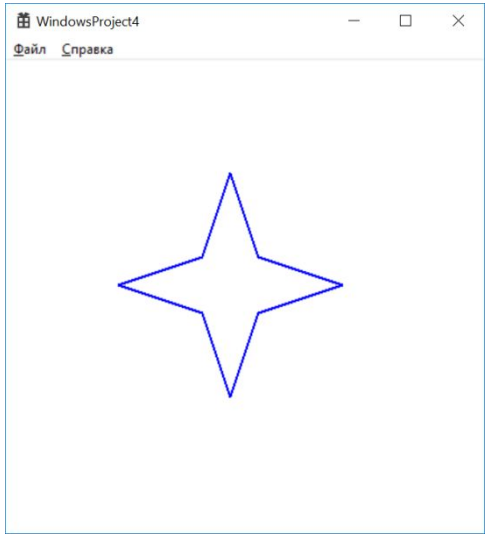


Image4

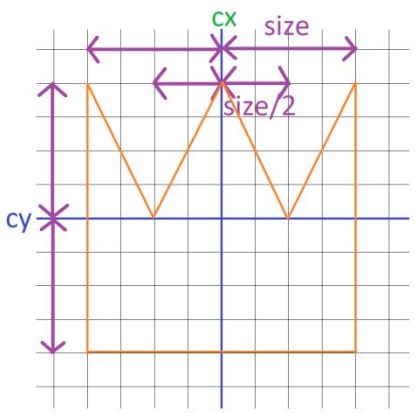
```
void Image4(HDC hdc, int cx, int cy, int size) {

    int x1 = cx;
    int y1 = cy - size;

    int x2 = cx + size / 4;
    int y2 = cy - size / 4;

    int x3 = cx + size;
    int y3 = cy;

    int x4 = cx + size / 4;
    int y4 = cy + size / 4;

    int x5 = cx;
    int y5 = cy + size;

    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(r_g, g_g, b_g));
    SelectObject(hdc, hPen);

    MoveToEx(hdc, x1, y1, NULL);
    LineTo(hdc, x2, y2);
    LineTo(hdc, x3, y3);
    LineTo(hdc, x4, y4);
    LineTo(hdc, x5, y5);
    LineTo(hdc, x6, y6);
    LineTo(hdc, x7, y7);
    LineTo(hdc, x8, y8);
    LineTo(hdc, x1, y1);

    DeleteObject(hPen);
}
```

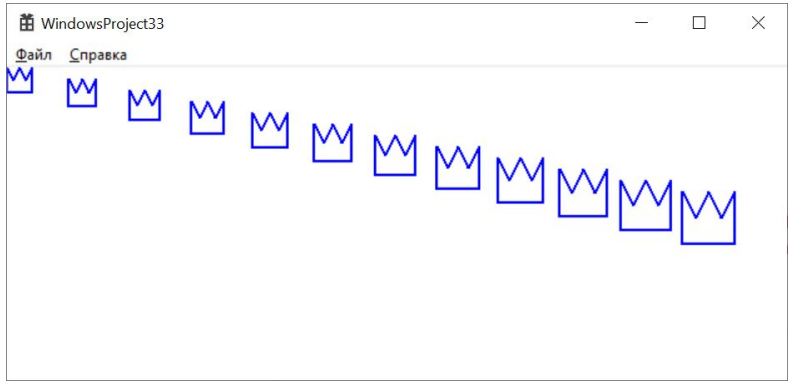


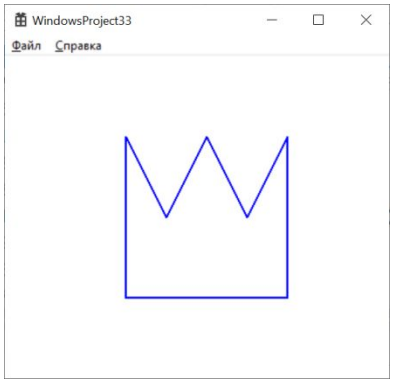

```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        Image4(hdc, 200, 200, 100);

        EndPaint(hWnd, &ps);
    }
    break;
```

Image 5 – Функция вызывает функцию

```
void Image5(HDC hdc, int cx, int cy, int size) {
    int x1 = cx;
    int y1 = cy - size;

    int x2 = cx + size / 2;
    int y2 = cy;

    int x3 = cx + size;
    int y3 = cy - size;

    int x4 = cx + size;
    int y4 = cy + size;

    int x5 = cx - size;
    int y5 = cy + size;
```



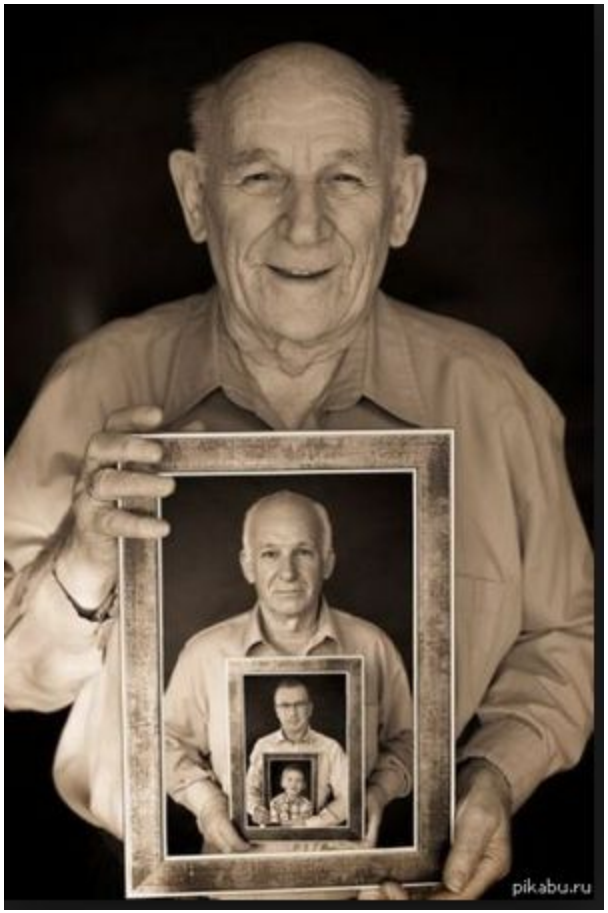





```
void task3(HDC hdc)
{
    int x = 10;
    int y = 10;
    int size = 10;

    do {

        Image5(hdc, x, y, size);

        x += 50;
        y += 10;
        size += 1;

    } while (x <= 600);
}
```

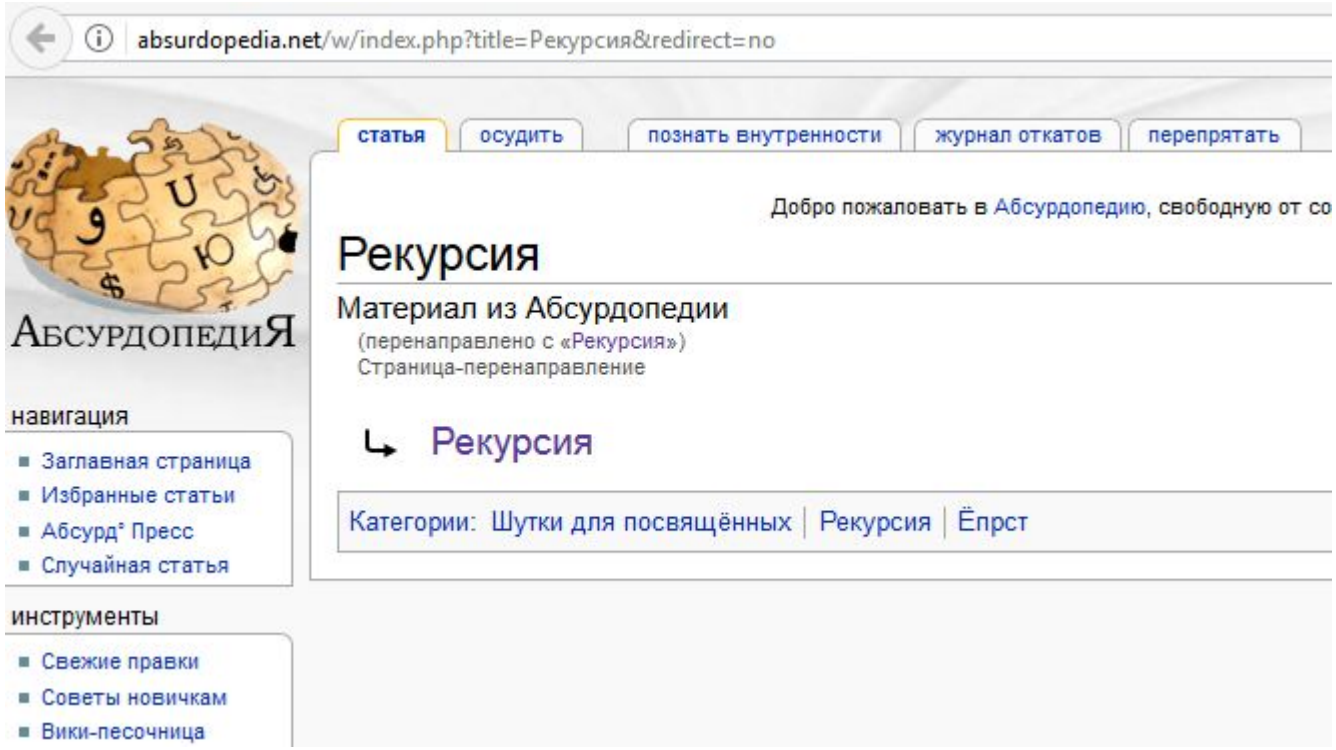
```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
    //example1(hdc);

    //task1(hdc);
    //task2(hdc);
    //task2_2(hdc);
    task3(hdc);
    //task3_2(hdc);
    //task3_3(hdc);

    //hometask1(hdc);
    //hometask2(hdc);
    //hometask3(hdc);
    //hometask4__(hdc);
    //hometask5__(hdc);
    EndPaint(hWnd, &ps);
}
break;
```

Рекурсия



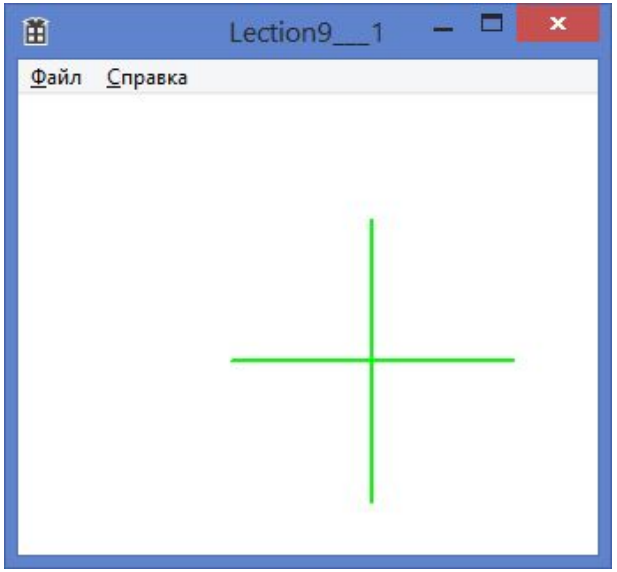
Рекурсия

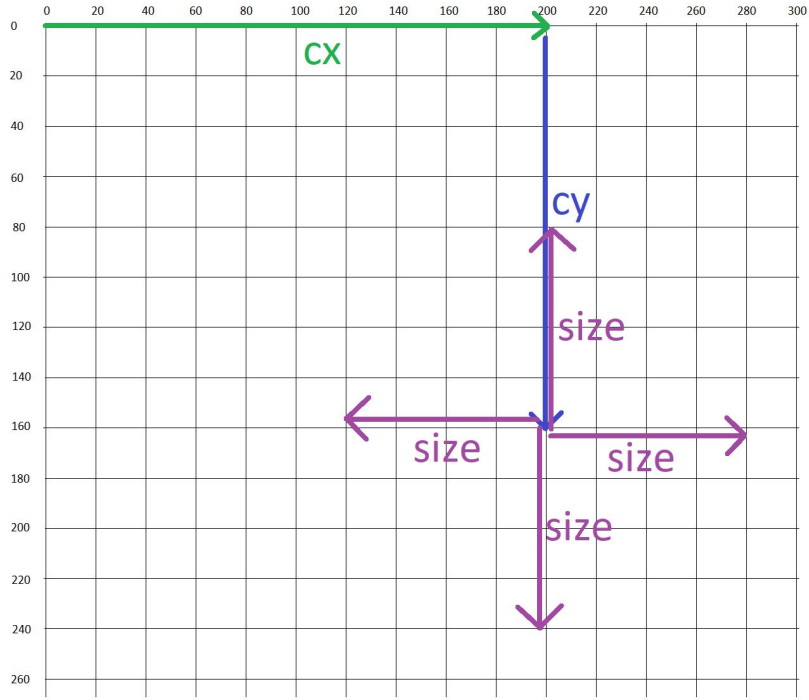
Рекурсия


absurdopedia.net/w/index.php?title=Рекурсия&redirect=no
АБСУРДОПЕДИЯ
статья
осудить
познать внутренности
журнал откатов
перепрятать
Добро пожаловать в Абсурдопедию, свободную от со
Рекурсия
Материал из Абсурдопедии
(перенаправлено с «Рекурсия»)
Страница-перенаправление
Рекурсия
Категории: Шутки для посвящённых | Рекурсия | Ёпрст
навигация
Заглавная страница
Избранные статьи
Абсурд° Пресс
Случайная статья
инструменты
Свежие правки
Советы новичкам
Вики-песочница

Рисование креста

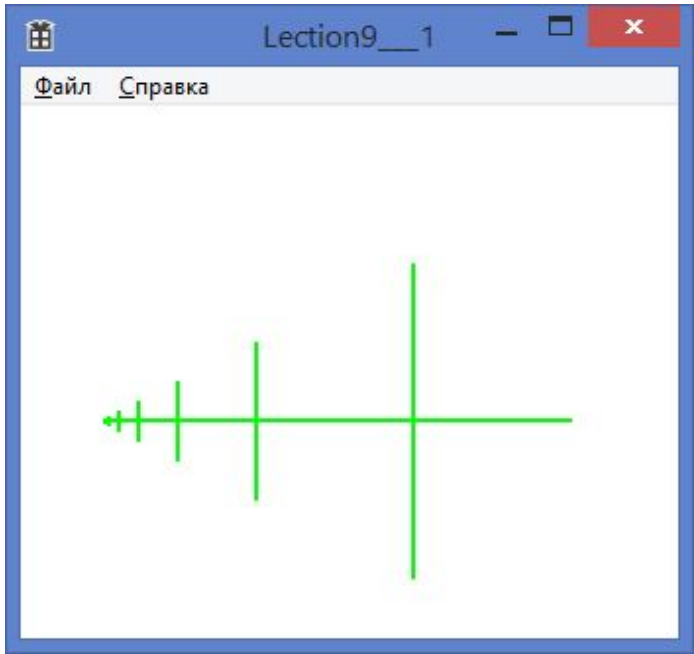
```
void Cross(HDC hdc, int cx, int cy, int size) {
        HPEN hPen;
        hPen = CreatePen(PS_SOLID, 2, RGB(0, 255, 0));
        SelectObject(hdc, hPen);
        MoveToEx(hdc, cx - size, cy, NULL);
        LineTo(hdc, cx + size, cy);
        MoveToEx(hdc, cx, cy - size, NULL);
        LineTo(hdc, cx, cy + size);
        DeleteObject(hPen);
}
 …
 case WM_PAINT:
      {
         PAINTSTRUCT ps;
         HDC hdc = BeginPaint(hWnd, &ps);
         // TODO: Добавьте сюда любой код прорисов
         Cross(hdc, 200, 160, 80);
         EndPaint(hWnd, &ps);
      }
      break;
```





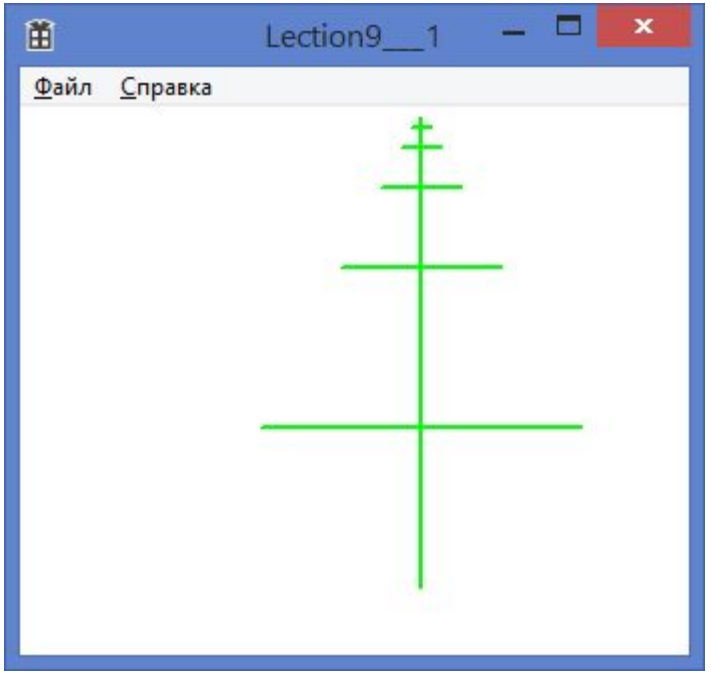
Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
      Cross(hdc, cx, cy, size);
      if (size < 2) {
            return;
      }
      RecursiveCross(hdc, cx - size, cy, size / 2);
}
…
 case WM_PAINT:     {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hWnd, &ps);

      RecursiveCross(hdc, 200, 160, 80);

      EndPaint(hWnd, &ps);
       }
```



Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }

    RecursiveCross(hdc, cx, cy - size, size / 2);
}
…

RecursiveCross(hdc, 200, 160, 80);
```



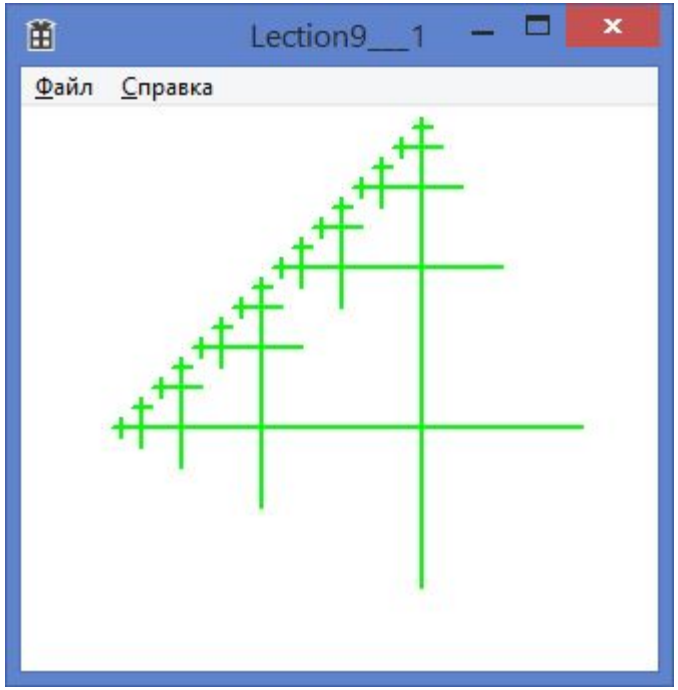
Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCross(hdc, cx - size, cy, size / 2);
    RecursiveCross(hdc, cx, cy - size, size / 2);
}…

RecursiveCross(hdc, 200, 160, 80);
```

?

Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCross(hdc, cx - size, cy, size / 2);
    RecursiveCross(hdc, cx, cy - size, size / 2);
}...

RecursiveCross(hdc, 200, 160, 80);
```



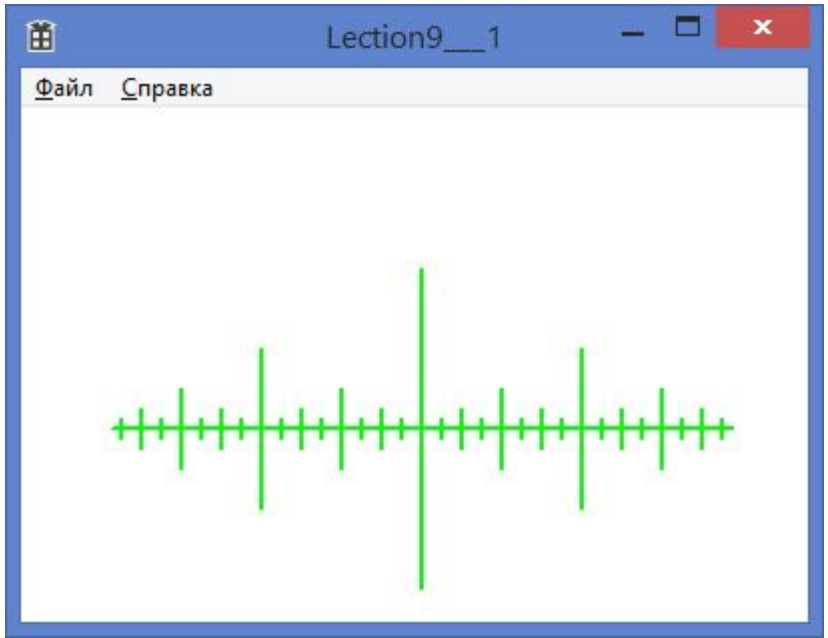
Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCross(hdc, cx - size, cy, size / 2);
    RecursiveCross(hdc, cx + size, cy, size / 2);
}…

RecursiveCross(hdc, 200, 160, 80);
```

?

Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCross(hdc, cx - size, cy, size / 2);
    RecursiveCross(hdc, cx + size, cy, size / 2);
}…

RecursiveCross(hdc, 200, 160, 80);
```



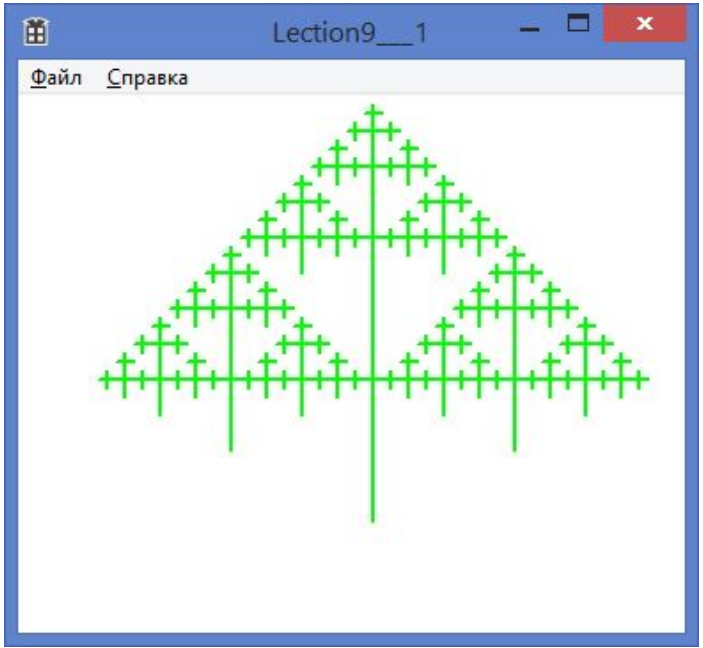
Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCross(hdc, cx - size, cy, size / 2);
    RecursiveCross(hdc, cx, cy - size, size / 2);
    RecursiveCross(hdc, cx + size, cy, size / 2);
}
...
RecursiveCross(hdc, 200, 160, 80);
```

?

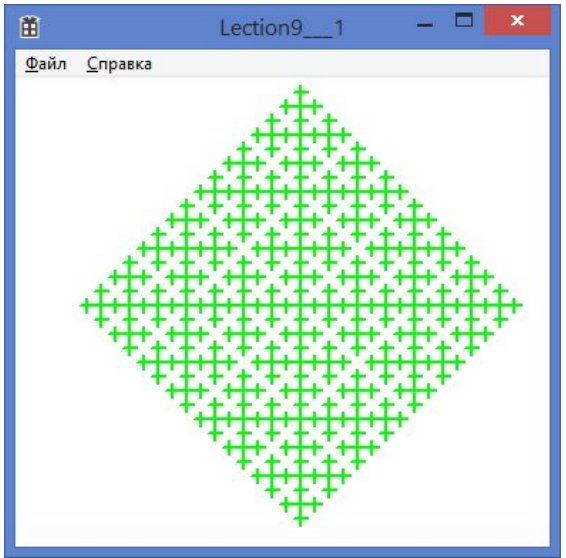
Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCross(hdc, cx - size, cy, size / 2);
    RecursiveCross(hdc, cx, cy - size, size / 2);
    RecursiveCross(hdc, cx + size, cy, size / 2);
}
...
RecursiveCross(hdc, 200, 160, 80);
```



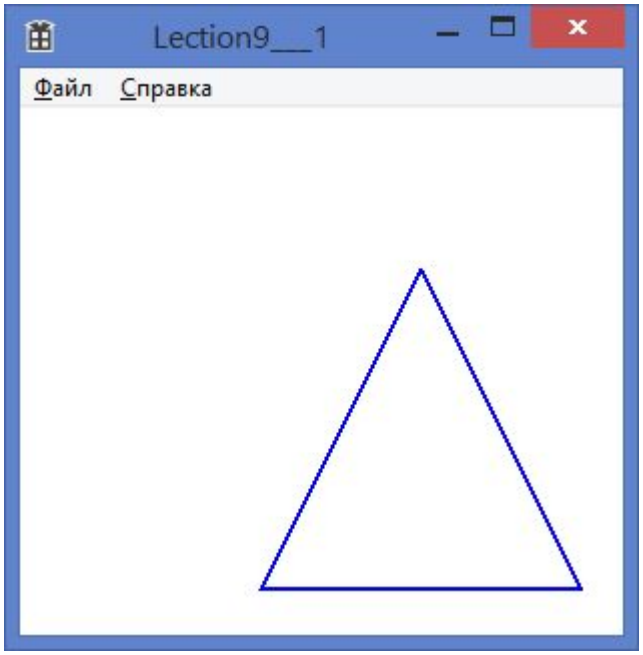
Рисование рекурсивного креста

```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {
    Cross(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
        RecursiveCross(hdc, cx - size, cy, size / 2);
        RecursiveCross(hdc, cx, cy - size, size / 2);
        RecursiveCross(hdc, cx + size, cy, size / 2);
        RecursiveCross(hdc, cx, cy + size, size / 2);
}
...
RecursiveCross(hdc, 200, 160, 80);
```



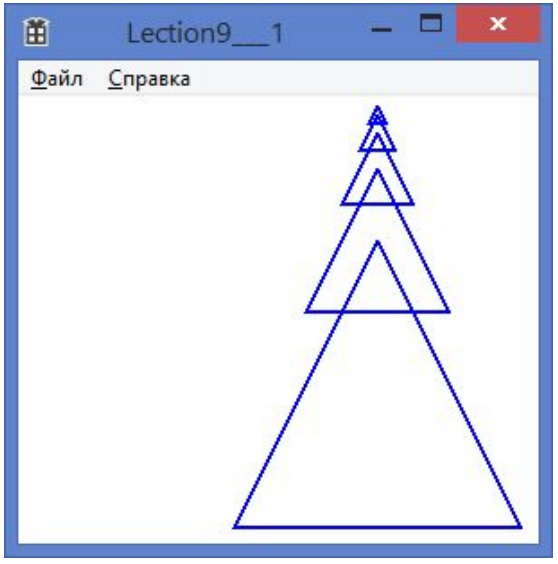
Рисование треугольника

```
void Triangle(HDC hdc, int cx, int cy, int size) {
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
    SelectObject(hdc, hPen);

    MoveToEx(hdc, cx, cy - size, NULL);
    LineTo(hdc, cx + size, cy + size);
    LineTo(hdc, cx - size, cy + size);
    LineTo(hdc, cx, cy - size);

    DeleteObject(hPen);
}
...
Triangle(hdc, 200, 160, 80);
```



Рисование рекурсивной фигуры с треугольником

```
void RecursiveTriangle(HDC hdc, int cx, int cy, int size) {
    Triangle(hdc, cx, cy, size);

    if (size < 10) {
        return;
    }

    RecursiveTriangle(hdc, cx, cy - size, size / 2);

}...
RecursiveTriangle(hdc, 200, 160, 80);
```



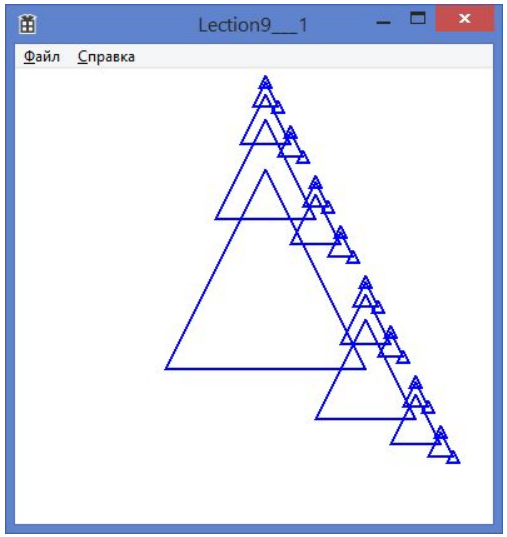
Рисование рекурсивной фигуры с треугольником

```
void RecursiveTriangle(HDC hdc, int cx, int cy, int size) {
    Triangle(hdc, cx, cy, size);

    if (size < 10) {
        return;
    }

    RecursiveTriangle(hdc, cx, cy - size, size / 2);
    RecursiveTriangle(hdc, cx + size, cy + size, size / 2);
}...
RecursiveTriangle(hdc, 200, 160, 80);
```

?

Рисование рекурсивной фигуры с треугольником

```
void RecursiveTriangle(HDC hdc, int cx, int cy, int size) {
    Triangle(hdc, cx, cy, size);

    if (size < 10) {
        return;
    }

    RecursiveTriangle(hdc, cx, cy - size, size / 2);
    RecursiveTriangle(hdc, cx + size, cy + size, size / 2);
}...
RecursiveTriangle(hdc, 200, 160, 80);
```

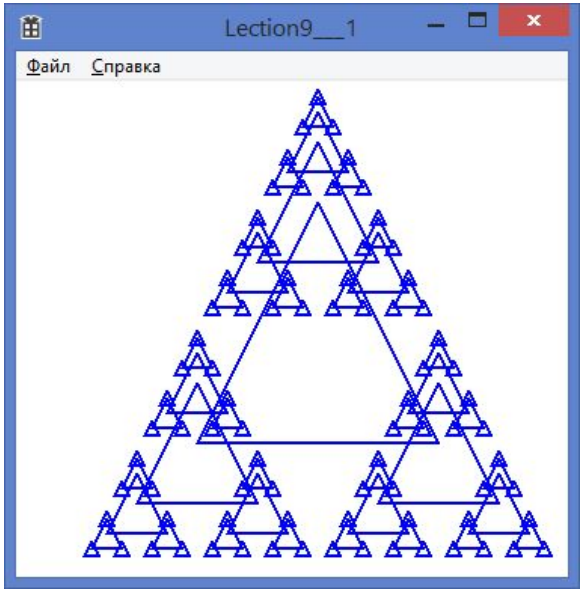


Рисование рекурсивной фигуры с треугольником

```
void RecursiveTriangle(HDC hdc, int cx, int cy, int size) {
    Triangle(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveTriangle(hdc, cx, cy - size, size / 2);
    RecursiveTriangle(hdc, cx + size, cy + size, size / 2);
    RecursiveTriangle(hdc, cx - size, cy + size, size / 2);
}
```

...

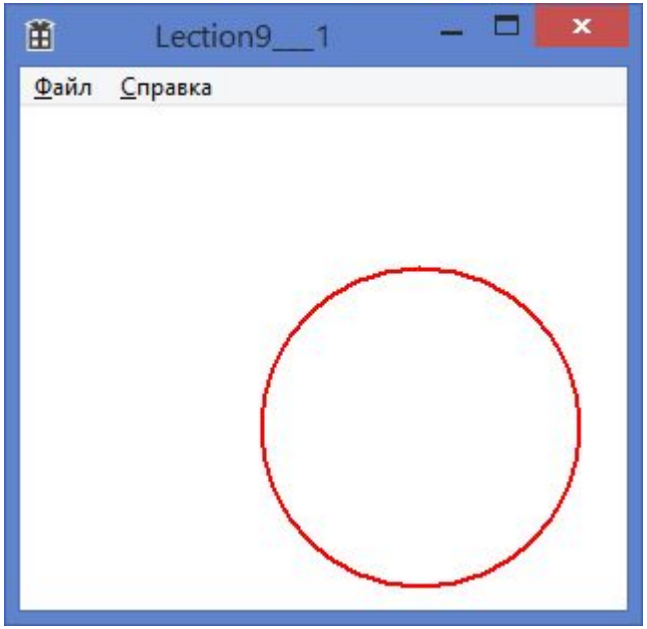
```
RecursiveTriangle(hdc, 200, 160, 80);
```



Рисование окружности

```
void Circle(HDC hdc, int cx, int cy, int size) {
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
    SelectObject(hdc, hPen);
    Ellipse(hdc, cx - size, cy - size, cx + size, cy + size);

    DeleteObject(hPen);
}
...
```

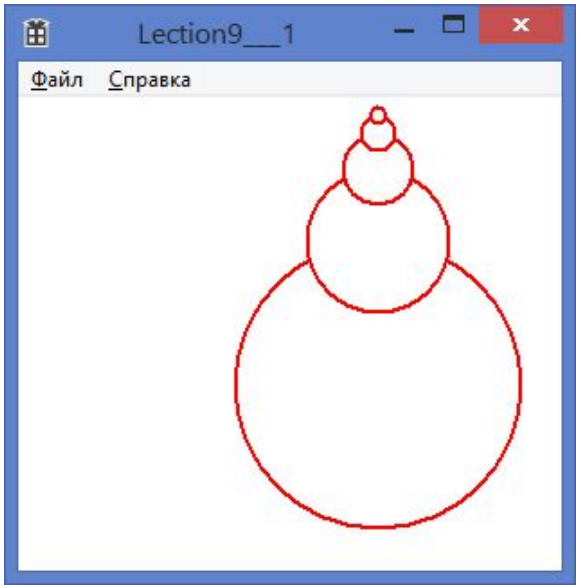
Circle(hdc, 200, 160, 80);



Рисование рекурсивной окружности

```
void RecursiveCircle(HDC hdc, int cx, int cy, int size) {
    Circle(hdc, cx, cy, size);

    if (size < 10) {
        return;
    }

    RecursiveCircle(hdc, cx, cy - size, size / 2);
}...
```

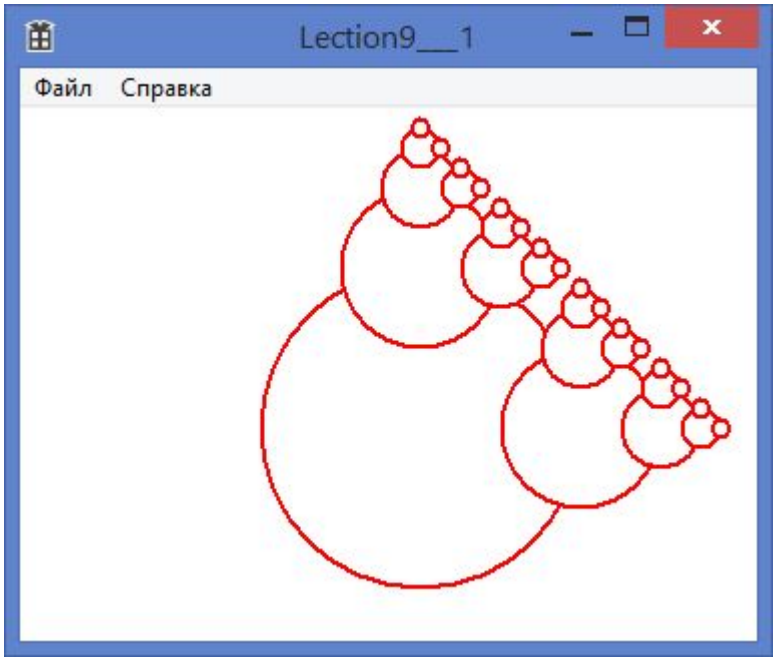
RecursiveCircle(hdc, 200, 160, 80);



Рисование рекурсивной окружности

```
void RecursiveCircle(HDC hdc, int cx, int cy, int size) {
    Circle(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCircle(hdc, cx, cy - size, size / 2);
    RecursiveCircle(hdc, cx + size, cy, size / 2);
}...
```

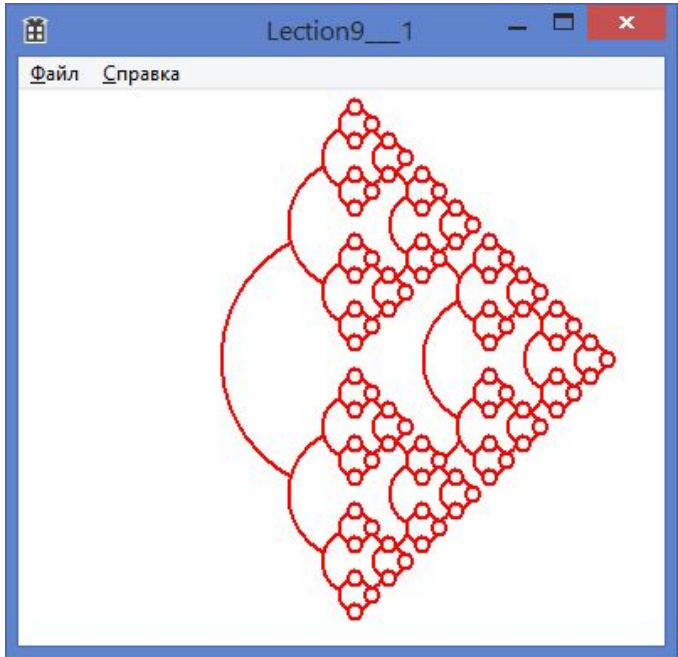
```
RecursiveCircle(hdc, 200, 160, 80);
```



Рисование рекурсивной окружности

```
void RecursiveCircle(HDC hdc, int cx, int cy, int size) {
    Circle(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCircle(hdc, cx, cy - size, size / 2);
    RecursiveCircle(hdc, cx + size, cy, size / 2);
    RecursiveCircle(hdc, cx, cy + size, size / 2);
}...
```

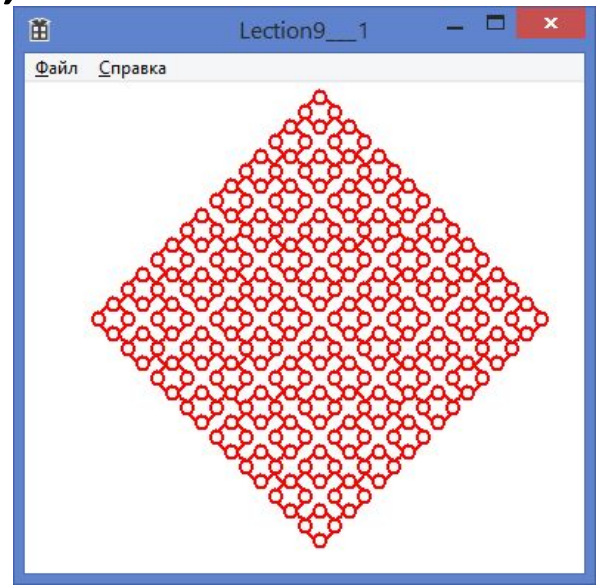
```
RecursiveCircle(hdc, 200, 160, 80);
```



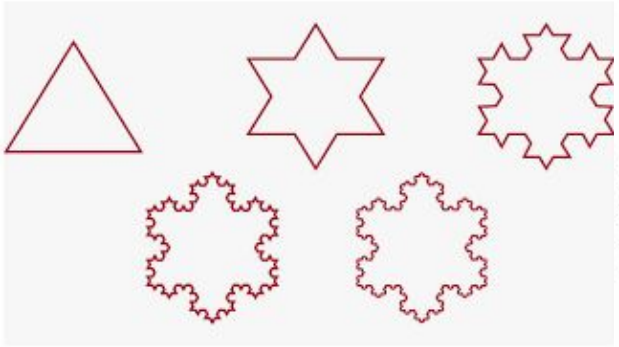
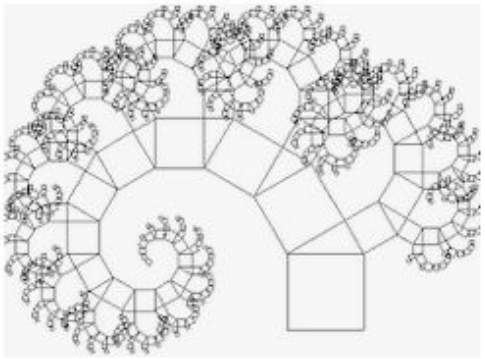
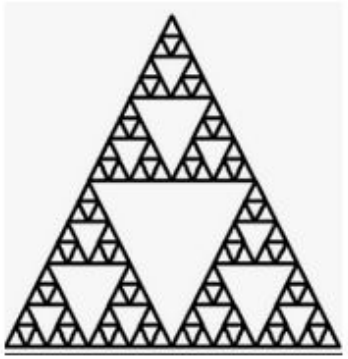
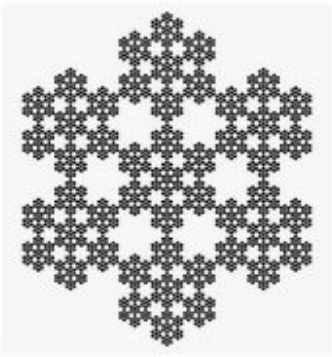
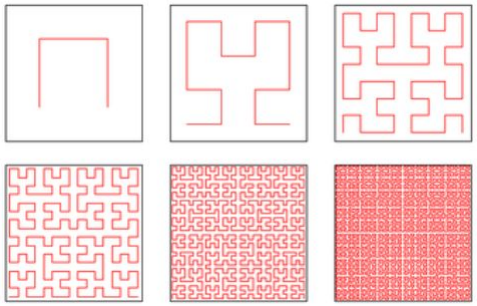
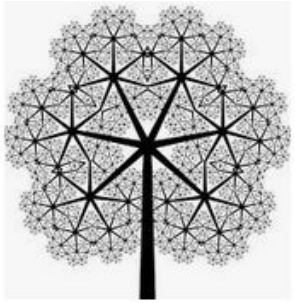
Рисование рекурсивной окружности

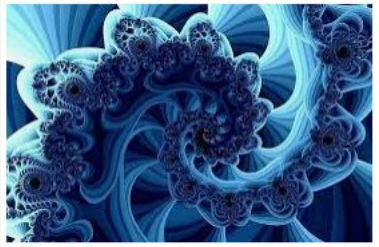
```
void RecursiveCircle(HDC hdc, int cx, int cy, int size) {
    Circle(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    RecursiveCircle(hdc, cx, cy - size, size / 2);
    RecursiveCircle(hdc, cx + size, cy, size / 2);
    RecursiveCircle(hdc, cx, cy + size, size / 2);
    RecursiveCircle(hdc, cx - size, cy, size / 2);
}...
```

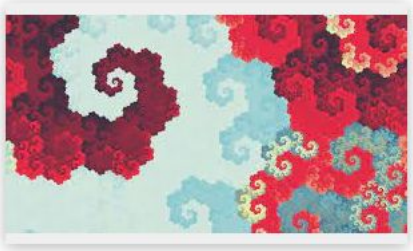
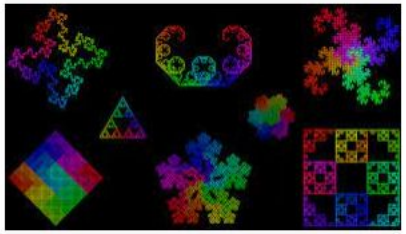
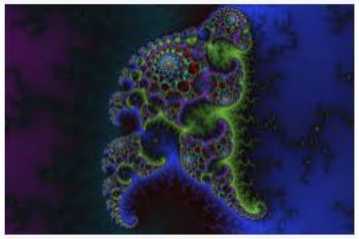
```
RecursiveCircle(hdc, 200, 160, 80);
```

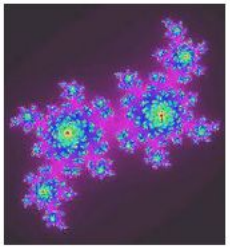


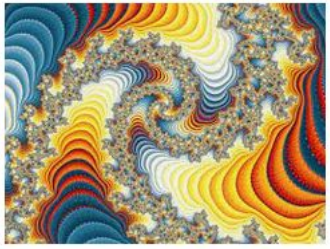
Фракталы



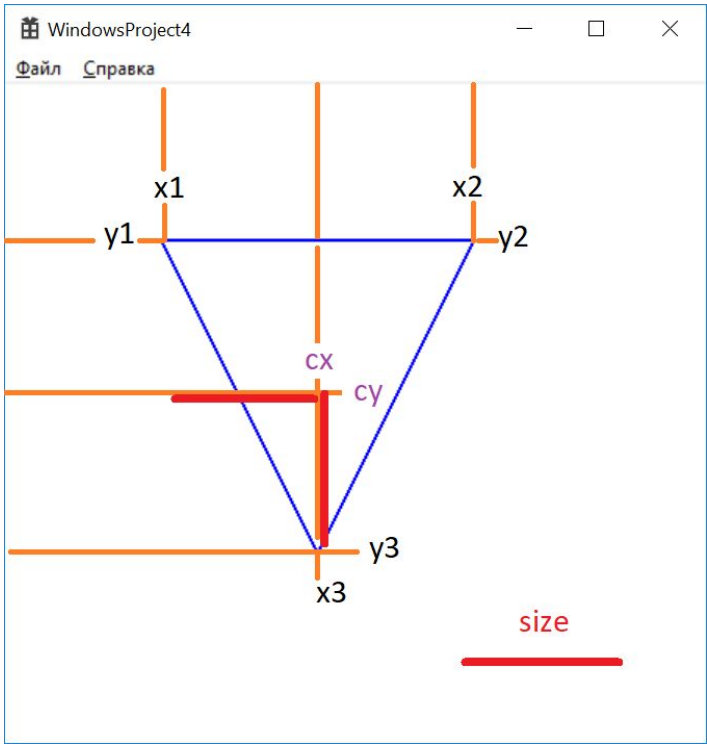








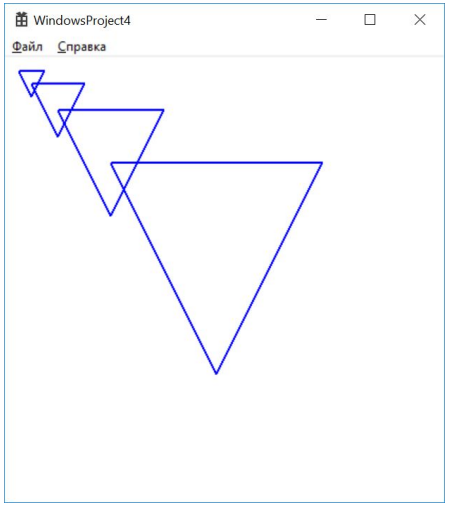

Рисование треугольника 2

```
void Triangle2(HDC hdc, int cx, int cy, int size) {
        int x1 = cx - size;
        int y1 = cy - size;

        int x2 = cx + size;
        int y2 = cy - size;

        int x3 = cx;
        int y3 = cy + size;

        HPEN hPen;
        hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
        SelectObject(hdc, hPen);
        MoveToEx(hdc, x1, y1, NULL);
        LineTo(hdc, x2, y2);
        LineTo(hdc, x3, y3);
        LineTo(hdc, x1, y1);
        DeleteObject(hPen);
}
…
Triangle2 (hdc, 200, 160, 80);
```

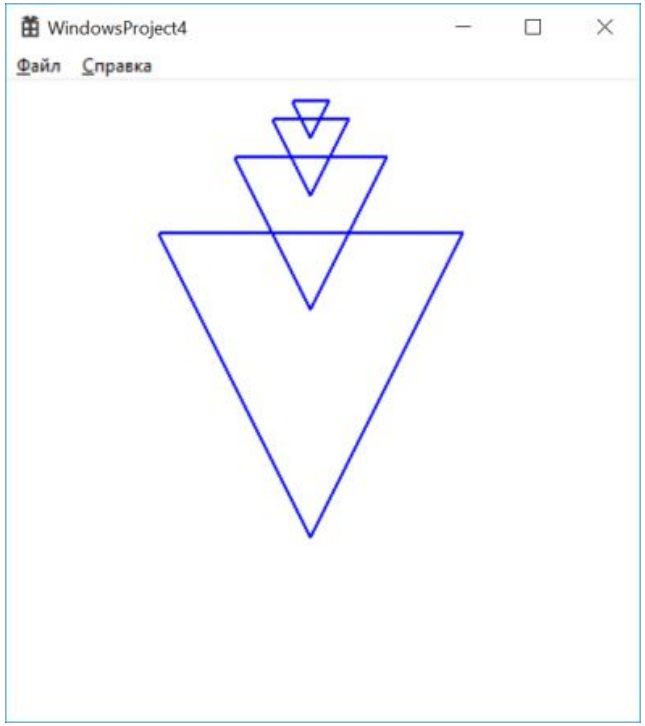


Рисование рекурсивной фигуры

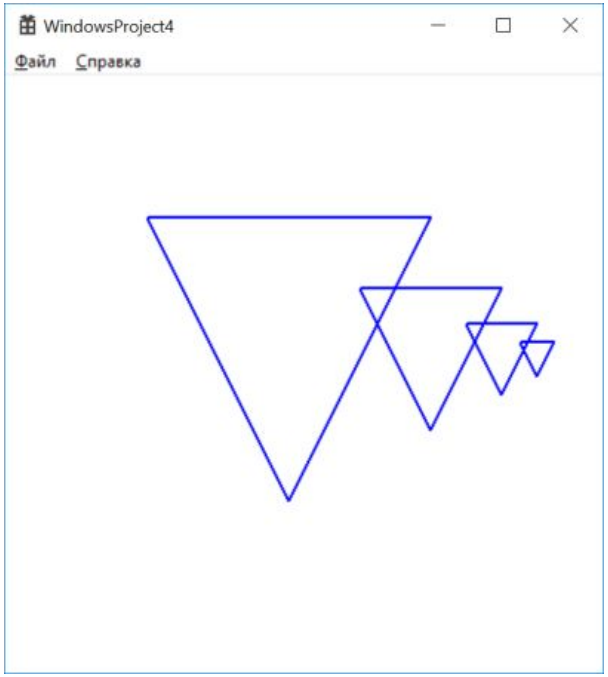
```
void RecursiveImage1_1(HDC hdc, int cx, int cy, int size) {
    Triangle2(hdc, cx, cy, size);
    if (size < 20) {
        return;
    }
    RecursiveImage1_1(hdc, cx - size, cy - size, size / 2);
}
...
 case WM_PAINT:    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);

    RecursiveImage1_1 (hdc, 200, 160, 80);

    EndPaint(hWnd, &ps);
     }
break;
```



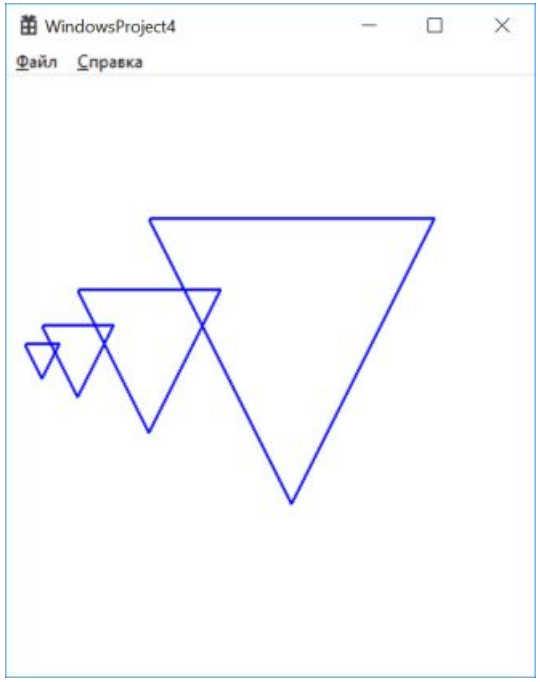
Задача 1.1. Нарисуйте



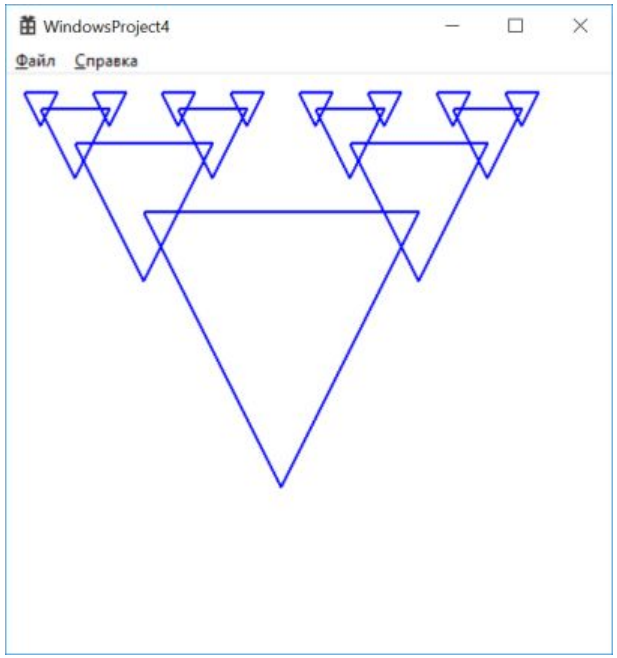
Задача 1.2. Нарисуйте



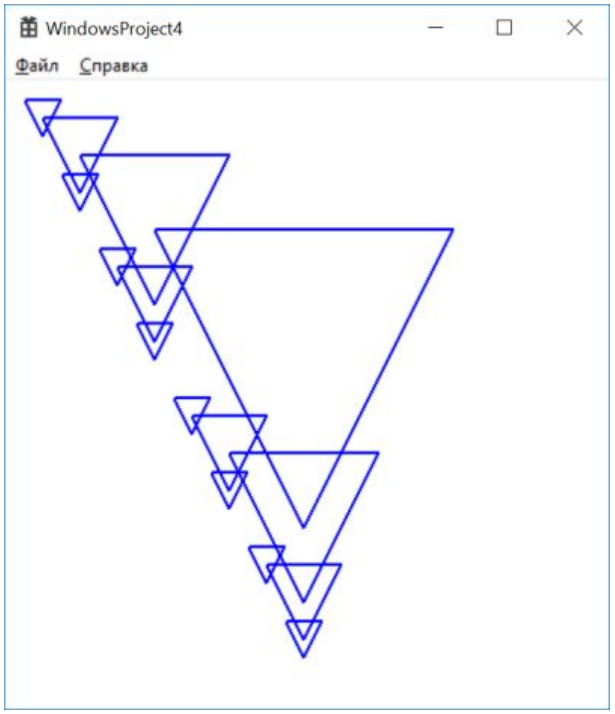
Задача 1.3. Нарисуйте



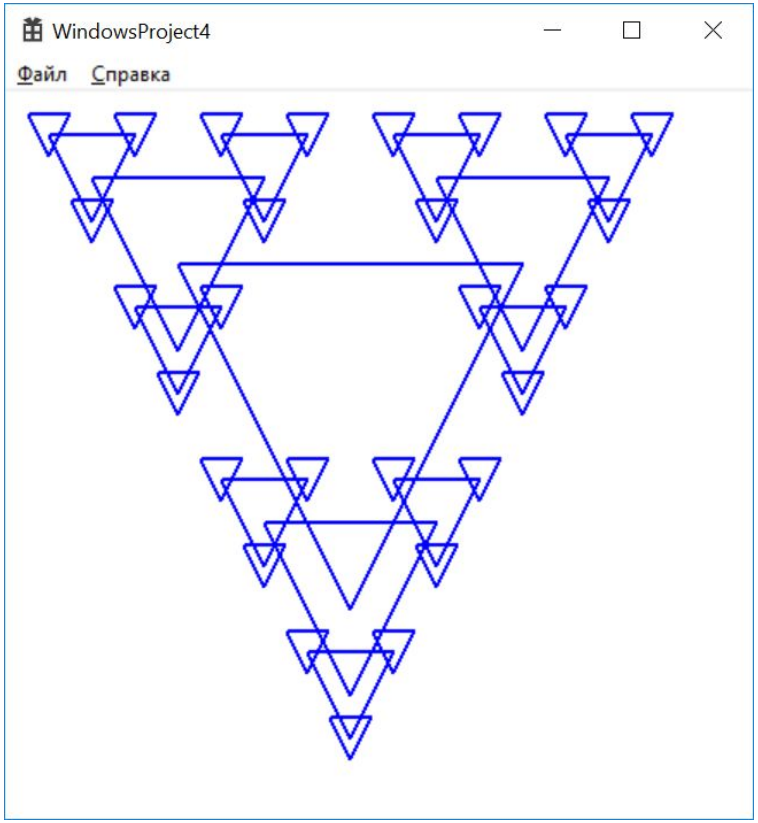
Задача 2.1. Нарисуйте



Задача 2.2. Нарисуйте



Задача 3.1.*. Нарисуйте



Задача 3.2*. Нарисуйте

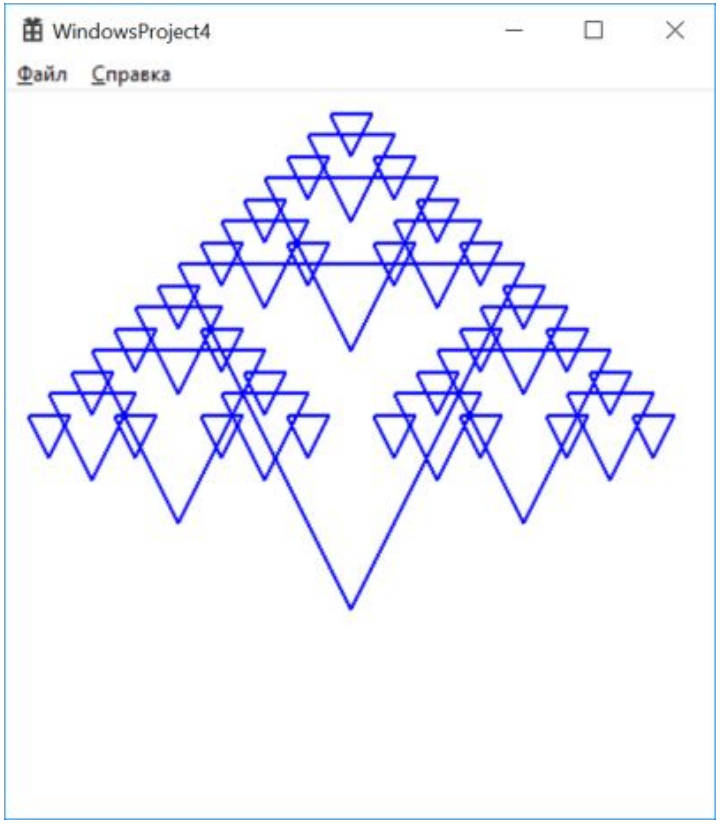
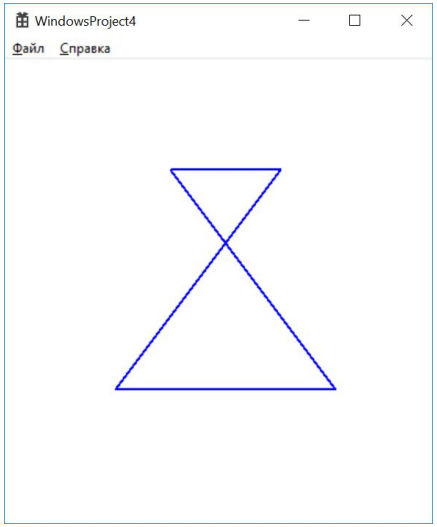


Image2

```
void Image2(HDC hdc, int cx, int cy, int size) {
    int x1 = cx - size / 2;
    int y1 = cy - size;
    int x2 = cx + size / 2;
    int y2 = cy - size;
    int x3 = cx - size;
    int y3 = cy + size;
    int x4 = cx + size;
    int y4 = cy + size;

    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
    SelectObject(hdc, hPen);

    MoveToEx(hdc, x1, y1, NULL);
    LineTo(hdc, x2, y2);
    LineTo(hdc, x3, y3);
    LineTo(hdc, x4, y4);
    LineTo(hdc, x1, y1);

    DeleteObject(hPen);
}
```



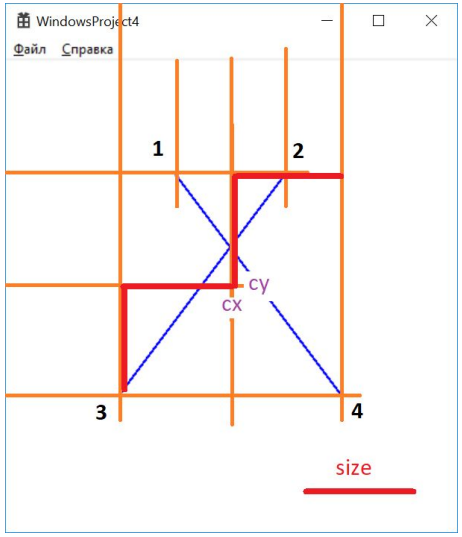
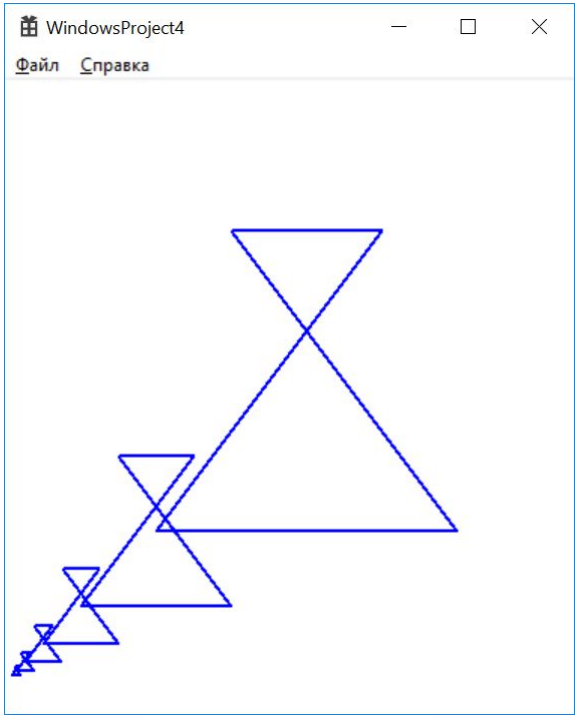


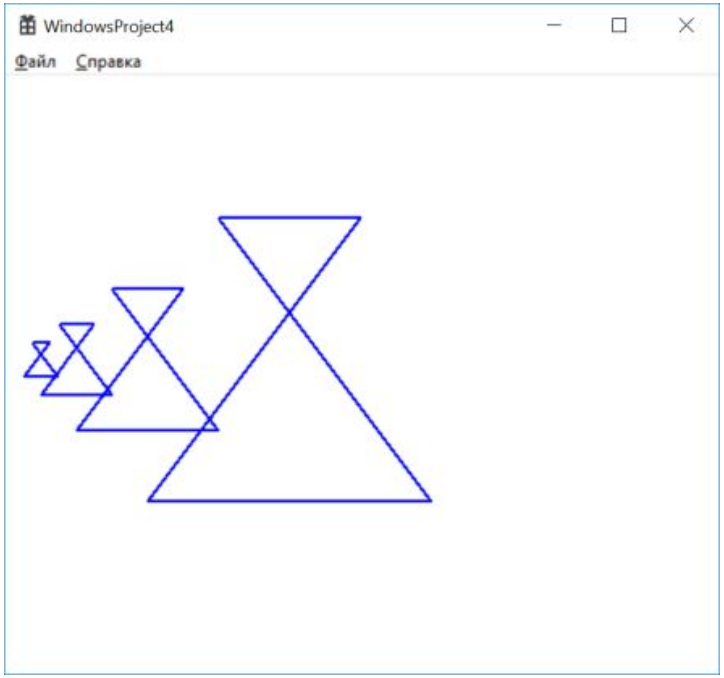
Image2 рекурсивный

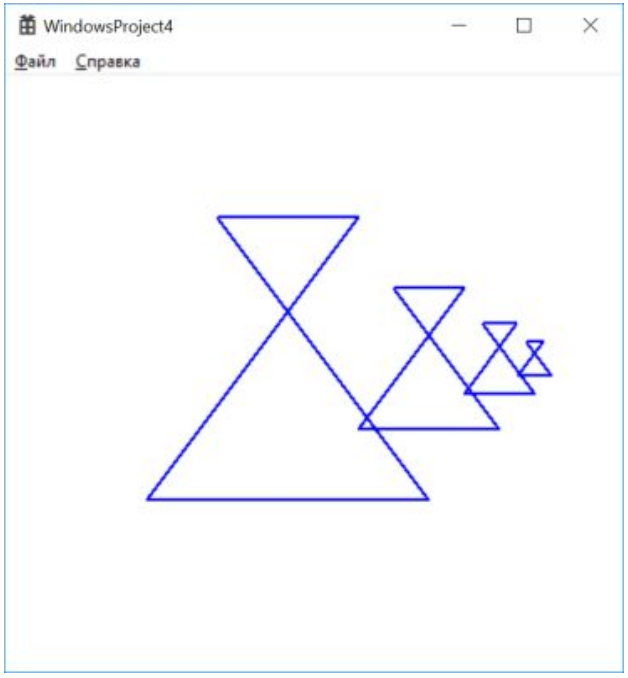
```
void RecursiveImage2_1(HDC hdc, int cx, int cy, int size) {
    Image2(hdc, cx, cy, size);
    if (size < 5) {
        return;
    }
    RecursiveImage2_1(hdc, cx - size , cy + size, size / 2);
}
```

```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        RecursiveImage2_1(hdc, 200, 200, 100);

        EndPaint(hWnd, &ps);
    }
    break;
```



Задача 4.1. - нарисуйте





Задача 4.2. - нарисуйте

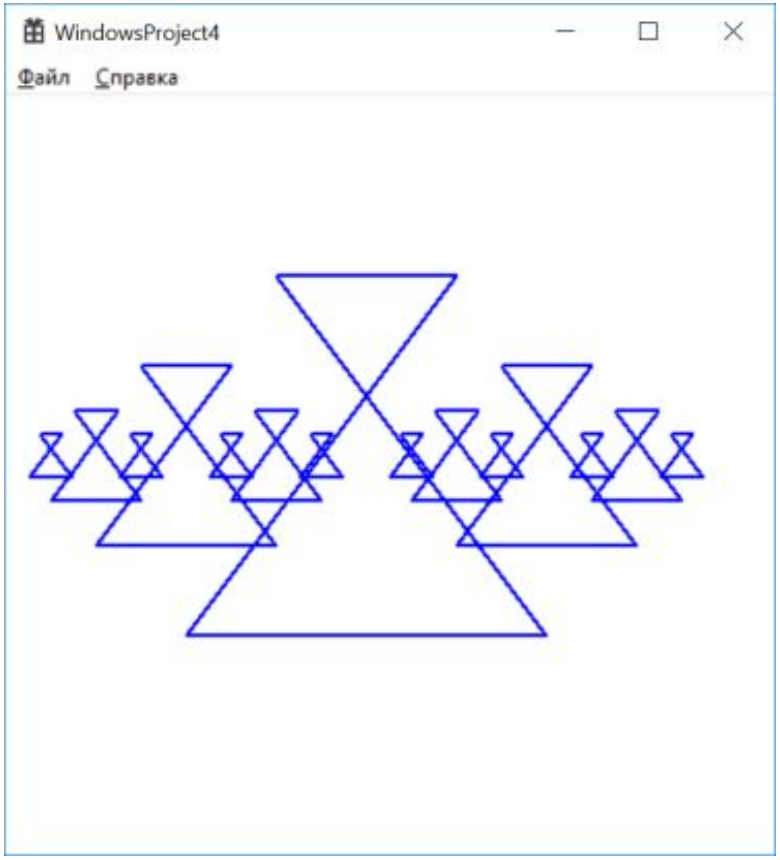
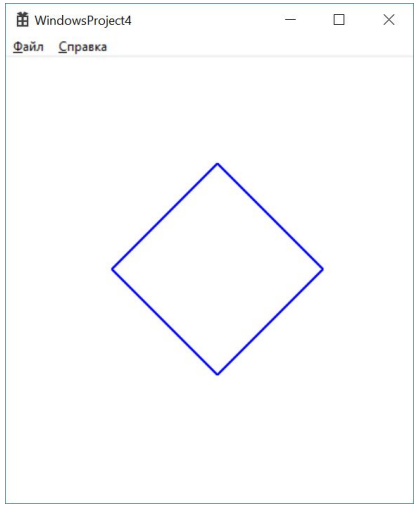


Image3

```
void Image3(HDC hdc, int cx, int cy, int size) {
    int x1 = cx;
    int y1 = cy - size;

    int x2 = cx + size;
    int y2 = cy;

    int x3 = cx;
    int y3 = cy + size;

    int x4 = cx - size;
    int y4 = cy;

    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
    SelectObject(hdc, hPen);

    MoveToEx(hdc, x1, y1, NULL);
    LineTo(hdc, x2, y2);
    LineTo(hdc, x3, y3);
    LineTo(hdc, x4, y4);
    LineTo(hdc, x1, y1);

    DeleteObject(hPen);
}
```

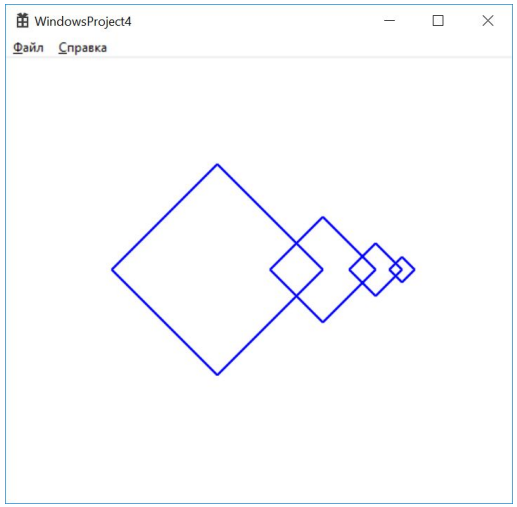



```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        Image3(hdc, 200, 200, 100);

        EndPaint(hWnd, &ps);
    }
    break;
```

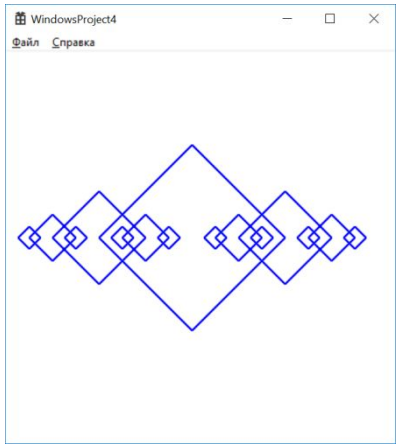
Image3 рекурсивный

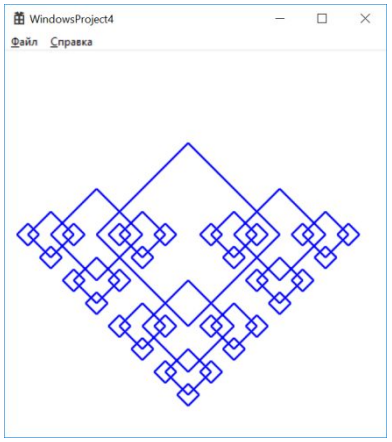
```
void RecursiveImage3_1(HDC hdc, int cx, int cy, int size) {
    Image3(hdc, cx, cy, size);
    if (size < 20) {
        return;
    }
    RecursiveImage3_1(hdc, cx + size, cy, size / 2);
}
```

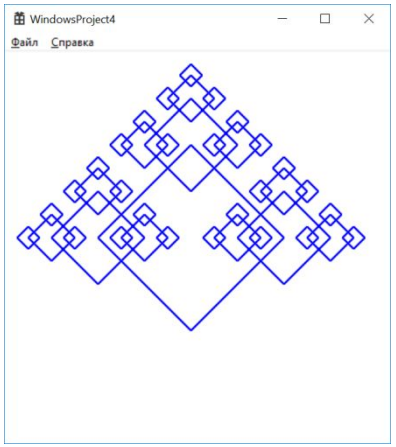
```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        RecursiveImage3_1(hdc, 200, 200, 100);

        EndPaint(hWnd, &ps);
    }
    break;
```

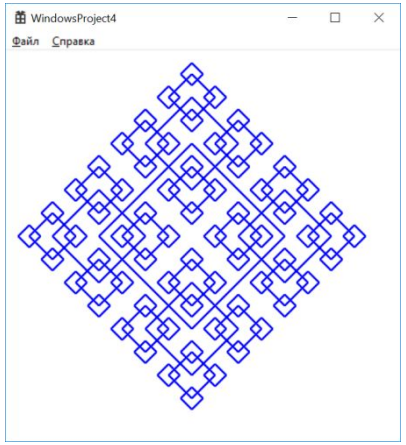


Задача 5* - нарисуйте

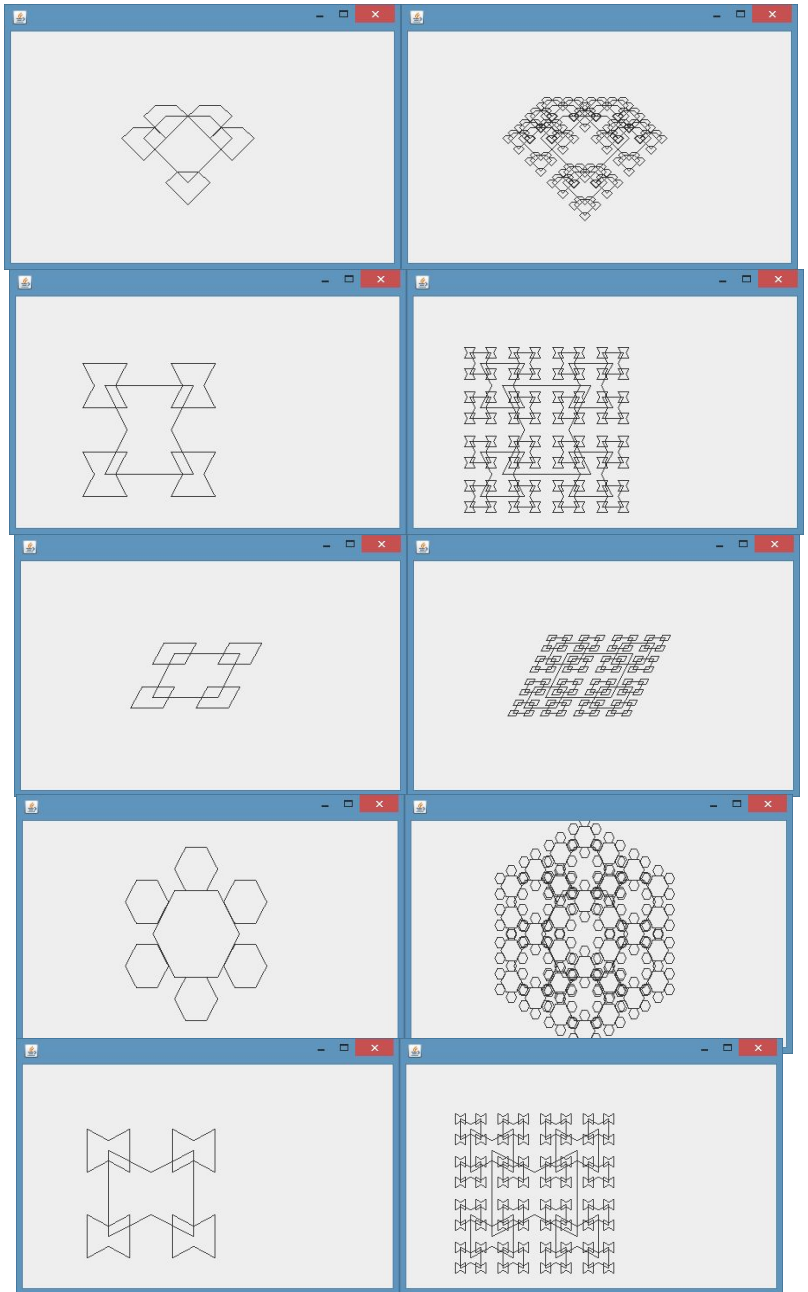


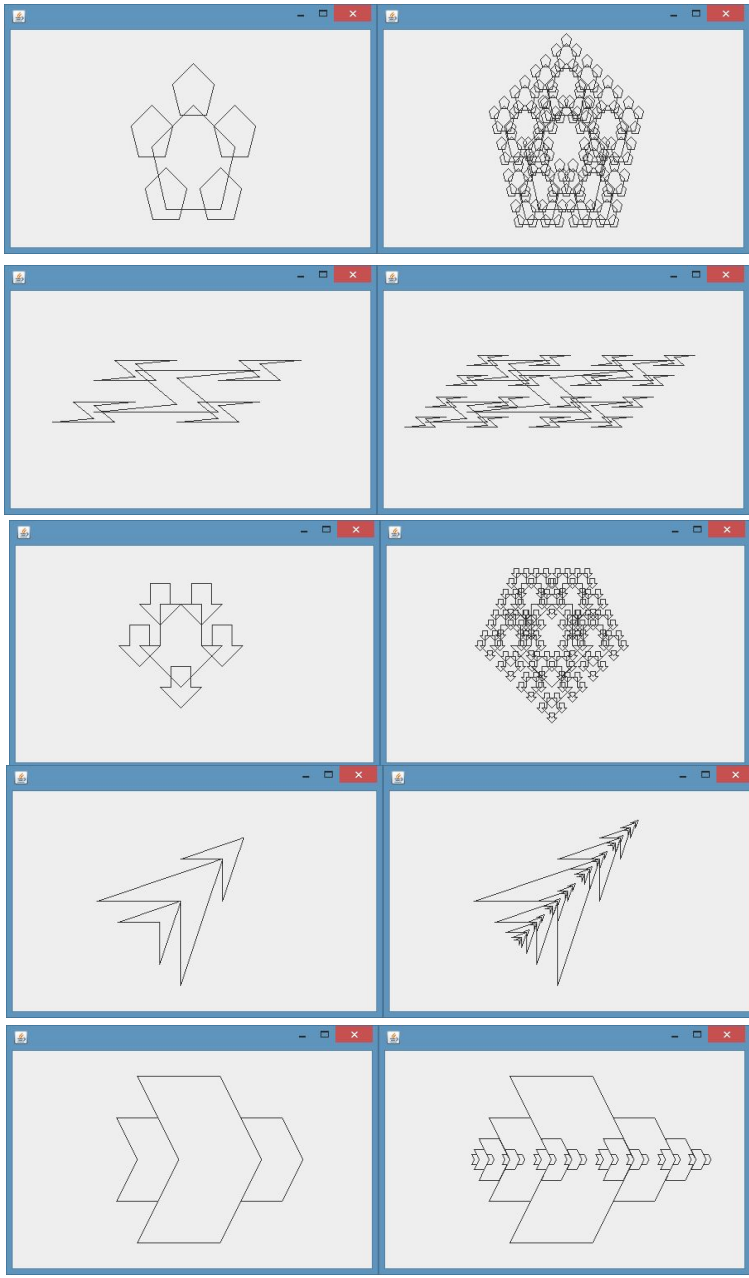






Варианты (Для вдохновения)





Домашнее задание

1) Выбрать 3 фигуры из лекции, вариантов на предыдущем слайде, или придумайте самостоятельно
2) Реализуйте функции

   MyImage1(HDC hdc, int cx, int cy, int size)

   MyImage2(HDC hdc, int cx, int cy, int size)

   MyImage3(HDC hdc, int cx, int cy, int size)

   которые отрисовывают выбранные вами фигуры.
3) Реализуйте функции

   MyRecursiveImage1_1(HDC hdc, int cx, int cy, int size)

   MyRecursiveImage1_2(HDC hdc, int cx, int cy, int size)

   MyRecursiveImage2_1(HDC hdc, int cx, int cy, int size)

   MyRecursiveImage2_2(HDC hdc, int cx, int cy, int size)

   MyRecursiveImage3_1(HDC hdc, int cx, int cy, int size)

   MyRecursiveImage3_2(HDC hdc, int cx, int cy, int size)

   которые отрисовывают рекурсивные картинки на основе MyImage1 – MyImage3 соответственно. Конкретные рекурсивные картинки – на ваш выбор!