

# Основы программирования

ФИСТ 1 курс

Власенко

Олег

Федосович

**Лекция 3**

Функции. Локальные переменные.

Передача параметров.

Рекурсия.

Рисование рекурсивных картинок

# Что такое подпрограмма?

*Процедуры и функции в Pascal.*

```
procedure ReadArray;  
begin  
end;
```

```
function Abs(x:single): single;  
begin  
end;
```

*Функции в Си.*

```
void read_array () {  
}
```

```
float abs(float f) {  
}
```

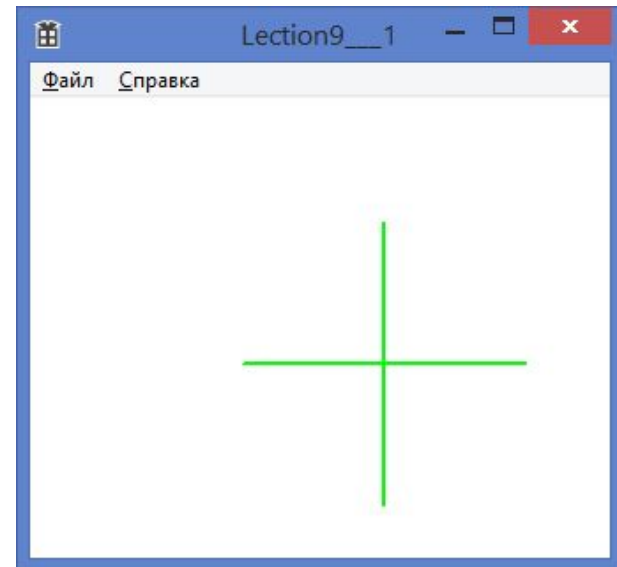
# **Зачем нужны подпрограммы?**

# Зачем нужны подпрограммы?

- Писать меньше кода - Повторяющийся код реализовать один раз, а вызывать многократно (sin(), printf() ...)
- Сделать код проще для редактирования - Разделить длинный код на части (произвольно)
- Упростить код - Разбить сложный алгоритм на части
- Повысить уровень абстракции – уйти от низкоуровневых операций на уровень предметной области
- Создавать библиотеки для повторного использования – стандартная библиотека Си состоит из функций
- Писать большие программы (до десятков и сотен

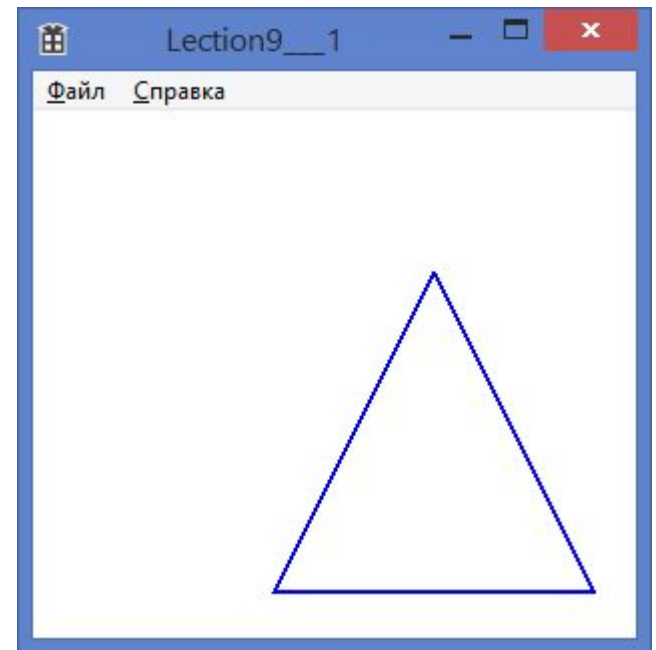
# Рисование креста

```
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);  
    Cross(hdc, 200, 150, 80);  
    EndPaint(hWnd, &ps);  
}
```



# Рисование треугольника

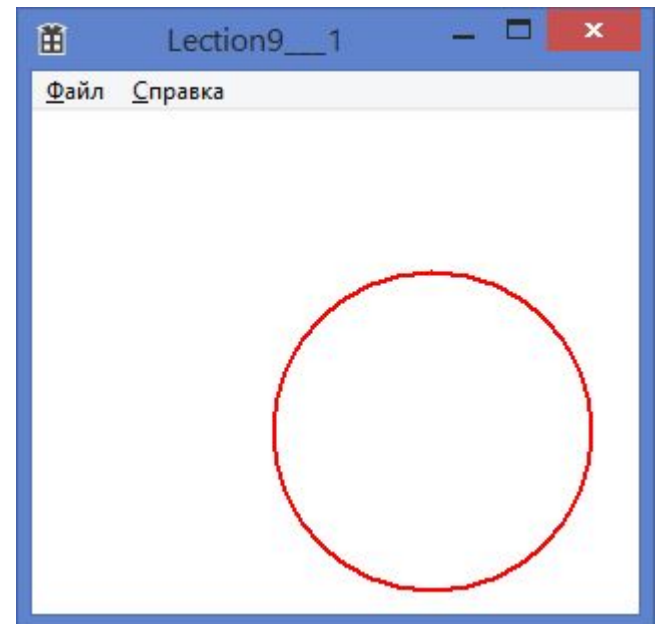
```
void Triangle(HDC hdc, int cx, int cy, int size) {  
    HPEN hPen;  
    hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));  
    SelectObject(hdc, hPen);  
  
    MoveToEx(hdc, cx, cy - size, NULL);  
    LineTo(hdc, cx + size, cy + size);  
    LineTo(hdc, cx - size, cy + size);  
    LineTo(hdc, cx, cy - size);  
  
    DeleteObject(hPen);  
}  
...  
Triangle(hdc, 200, 160, 80);
```



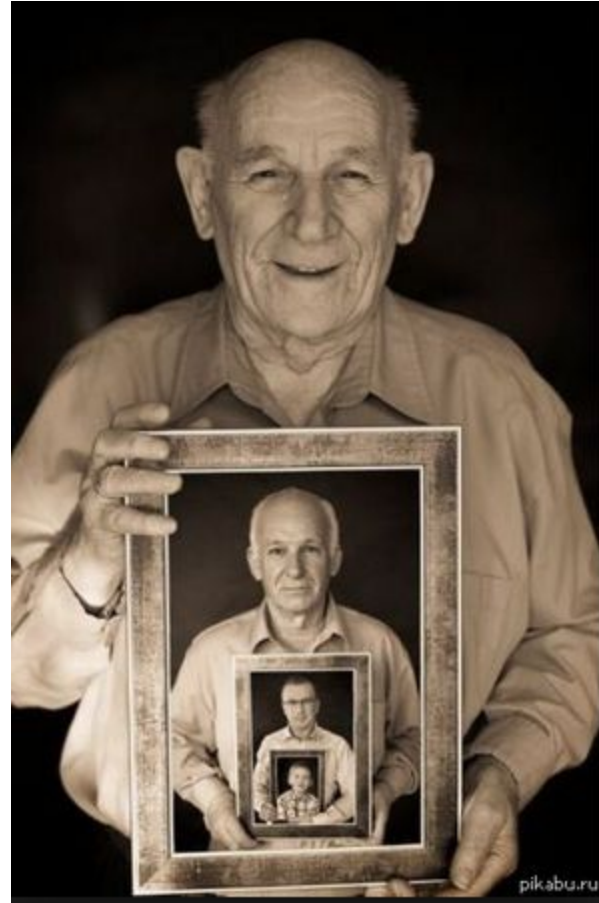
# Рисование окружности

```
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);
```



# Рекурсия






# Рекурсия



# Рекурсия

← ⓘ absurdopedia.net/w/index.php?title=Рекурсия&redirect=no



**АБСУРДОПЕДИЯ**

навигация

- [Заглавная страница](#)
- [Избранные статьи](#)
- [Абсурд° Пресс](#)
- [Случайная статья](#)

инструменты

- [Свежие правки](#)
- [Советы новичкам](#)
- [Вики-песочница](#)

[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](#)

# Рисование рекурсивного креста

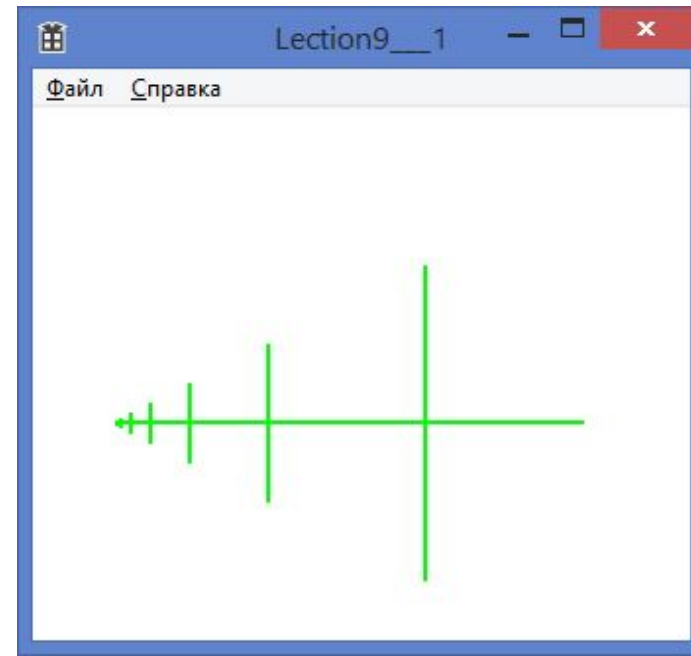
```
void RecursiveCross(HDC hdc, int cx, int cy, int size) {  
    Cross(hdc, cx, cy, size);  
    if (size < 2) {  
        return;  
    }  
    RecursiveCross(hdc, cx - size, cy, size / 2);  
}
```

...

```
case WM_PAINT: {  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);
```

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

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

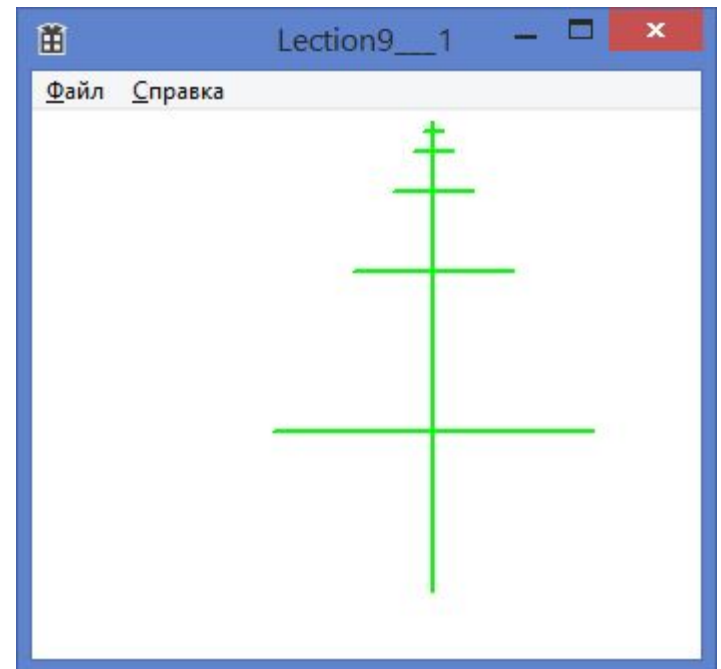


# Рисование рекурсивного креста

```
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);  
}...
```

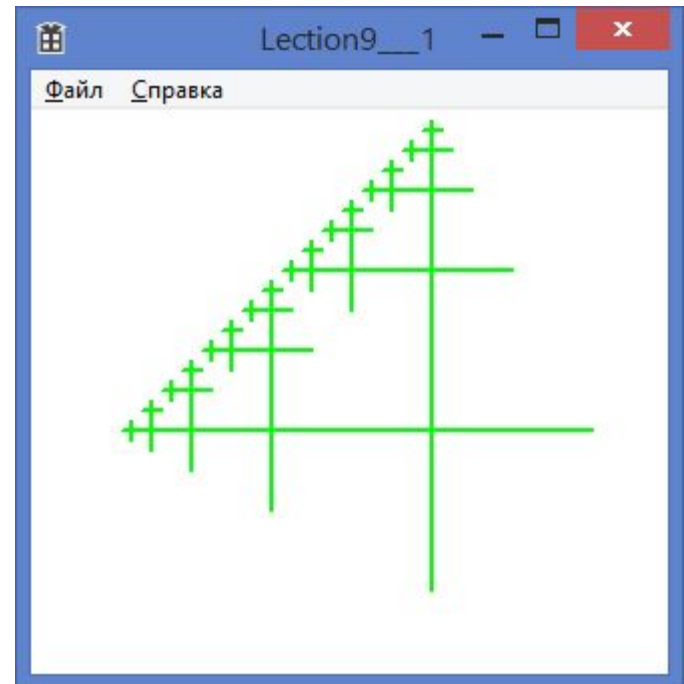
```
RecursiveCross(hdc, 200, 160, 80);
```

?

# 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);  
}...
```

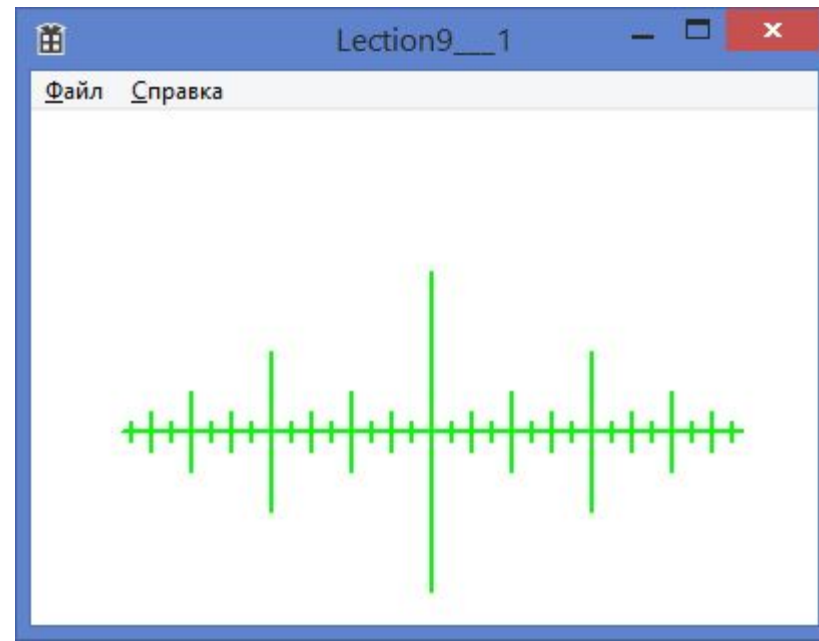
```
RecursiveCross(hdc, 200, 160, 80);
```

?

# 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);
```





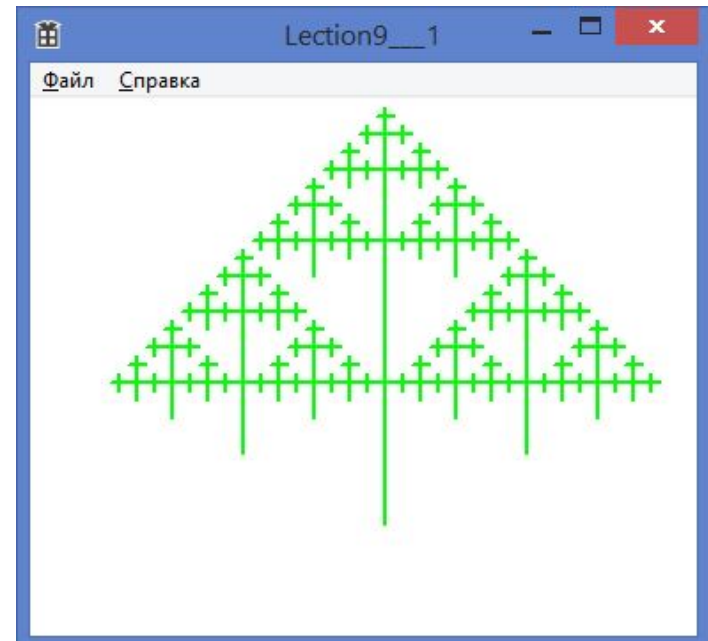
# Рисование рекурсивного креста

```
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);
```

?

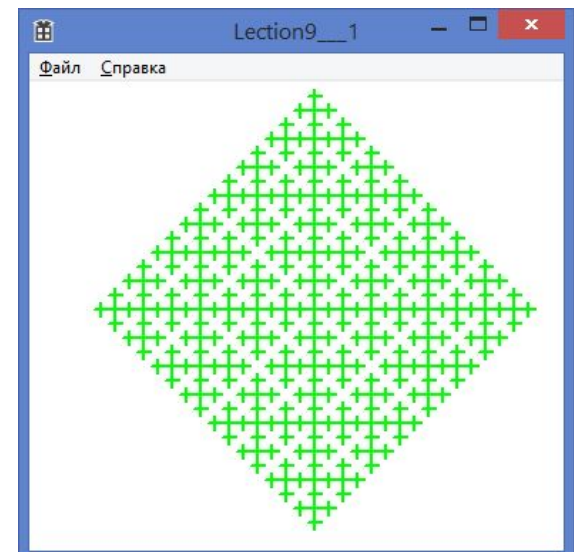
# Рисование рекурсивного креста

```
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);
```



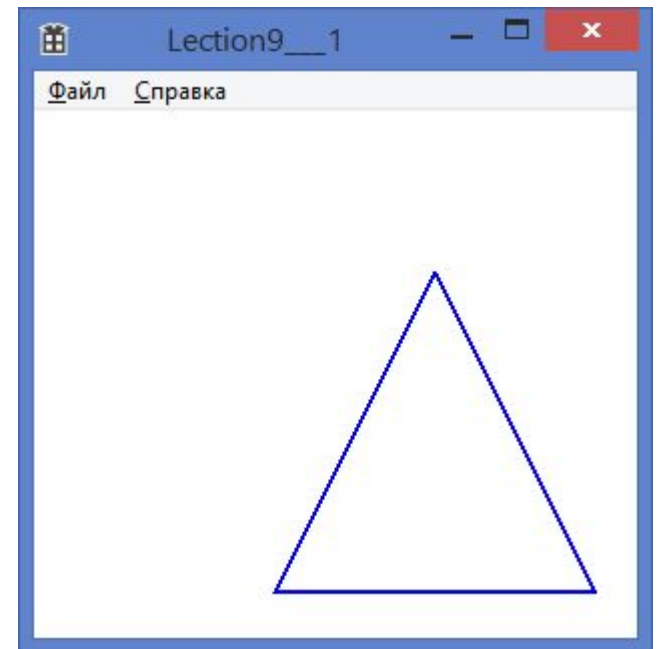
# Рисование рекурсивного креста

```
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

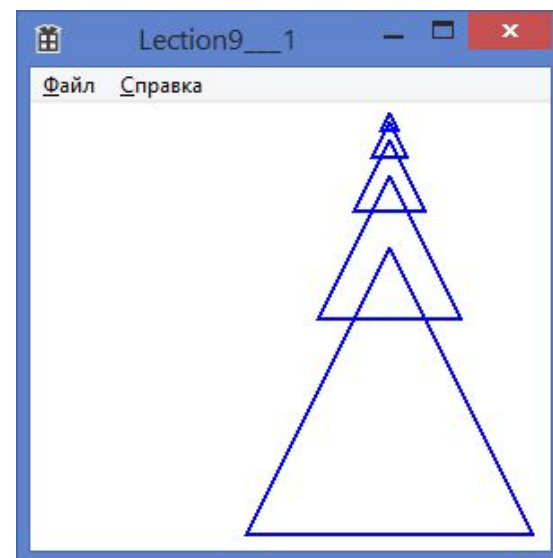
```
void RecursiveTriangle(HDC hdc, int cx, int cy, int size) {  
    Triangle(hdc, cx, cy, size);
```

```
    if (size < 10) {  
        return;  
    }
```

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

```
}...
```

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



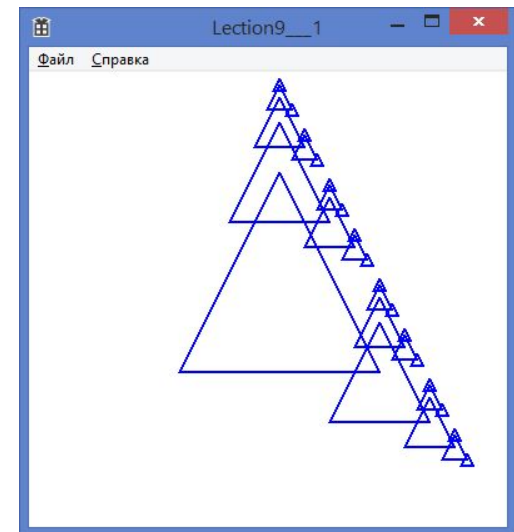
# Рисование рекурсивной фигуры с треугольником

```
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);
```

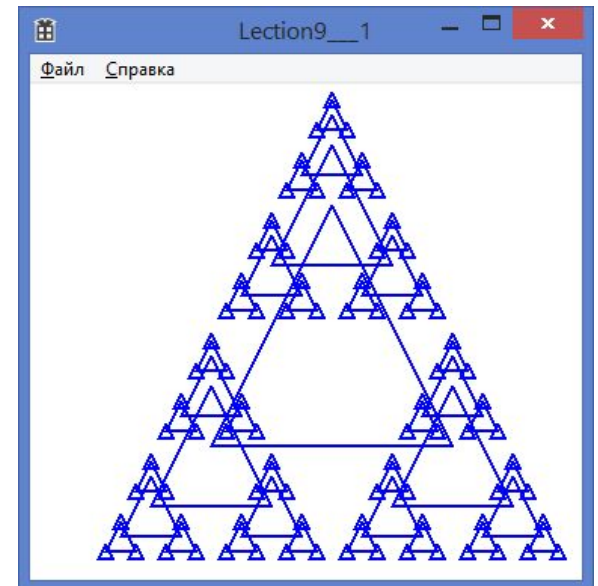


# Рисование рекурсивной фигуры с треугольником

```
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);
```

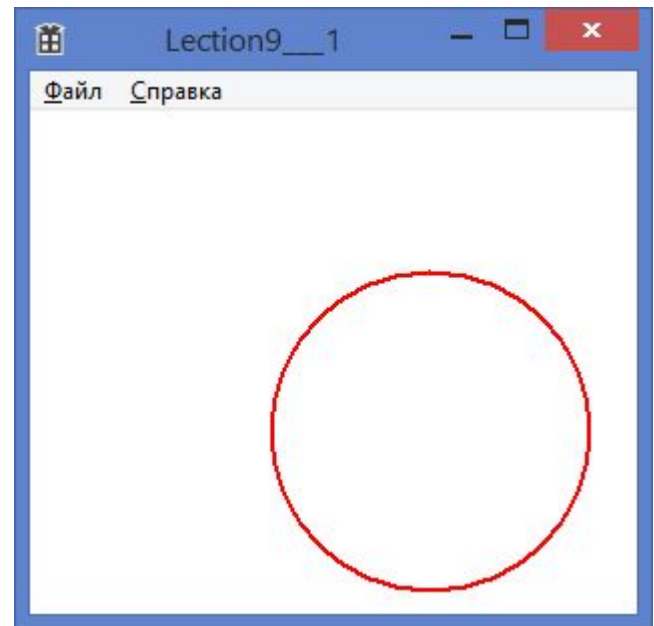




# Рисование окружности

```
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);  
}  
...
```

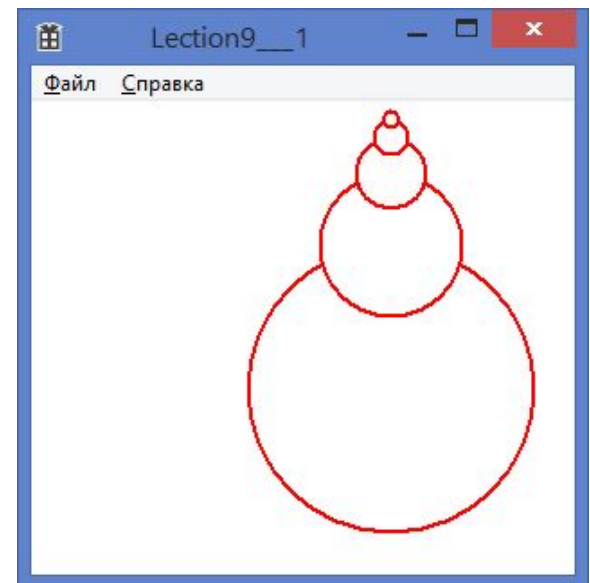
```
Circle(hdc, 200, 160, 80);
```



# 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);  
}...
```

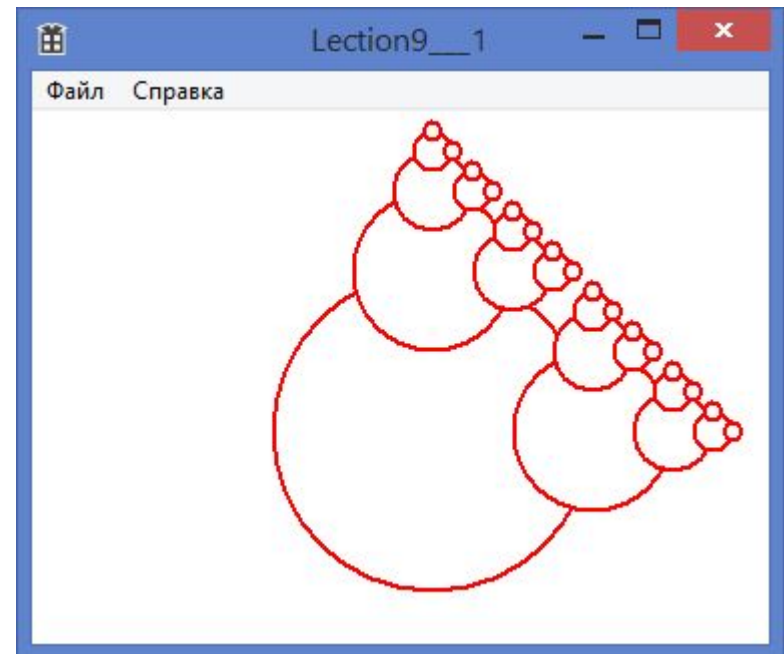
```
RecursiveCircle(hdc, 200, 160, 80);
```



# 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);  
}...
```

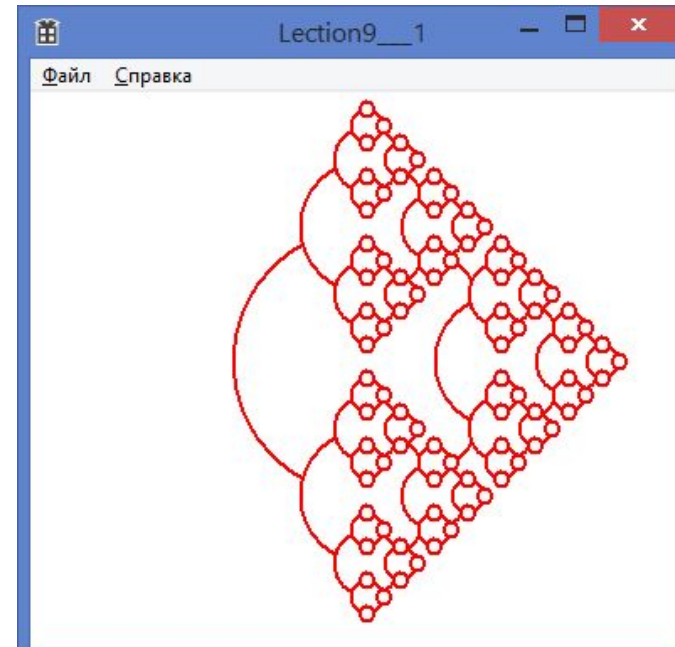
```
RecursiveCircle(hdc, 200, 160, 80);
```



# 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);  
}...
```

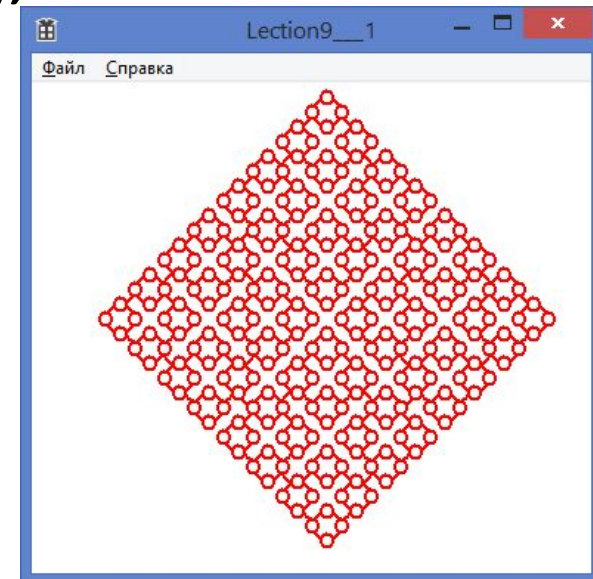
```
RecursiveCircle(hdc, 200, 160, 80);
```




# 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);
```



# Косвенная рекурсия



**АБСУРДОПЕДИЯ**

навигация

- [Заглавная страница](#)
- [Избранные статьи](#)
- [Абсурд° Пресс](#)
- [Случайная статья](#)

[статья](#) [осудить](#) [познать внутренности](#) [журнал откатов](#) [перепрятать](#)


Добро пожаловать в [Абсурдопедию](#), свободную от

## Бесконечная косвенная рекурсия

Материал из Абсурдопедии  
Страница-перенаправление

↳ [Косвенная рекурсия](#)

Категории: [Шутки для посвящённых](#) | [Рекурсия](#) | [Ёпрст](#)



**АБСУРДОПЕДИЯ**

навигация

- [Заглавная страница](#)
- [Избранные статьи](#)
- [Абсурд° Пресс](#)
- [Случайная статья](#)

[статья](#) [осудить](#) [познать внутренности](#) [журнал откатов](#) [перепрятать](#)

Добро пожаловать в [Абсурдопедию](#), свободную от

## Косвенная рекурсия

Материал из Абсурдопедии  
Страница-перенаправление

↳ [Бесконечная косвенная рекурсия](#)

Категории: [Шутки для посвящённых](#) | [Рекурсия](#) | [Ёпрст](#)

# Косвенная рекурсия (1)

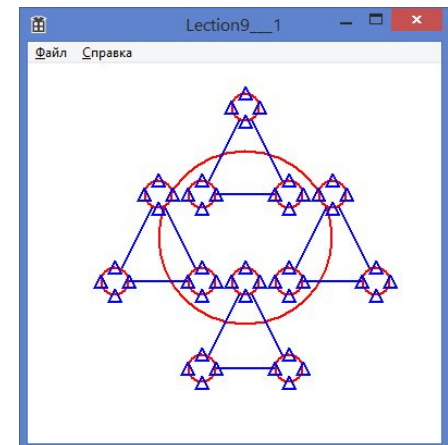
```
void IndirectRecursiveCircle(HDC hdc, int cx, int cy, int size);  
void IndirectRecursiveTriangle(HDC hdc, int cx, int cy, int size);
```

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

```
    if (size < 10) {  
        return;  
    }  
}
```

```
    IndirectRecursiveTriangle(hdc, cx, cy - size, size / 2);  
    IndirectRecursiveTriangle(hdc, cx + size, cy, size / 2);  
    IndirectRecursiveTriangle(hdc, cx, cy + size, size / 2);  
    IndirectRecursiveTriangle(hdc, cx - size, cy, size / 2);
```

```
}
```

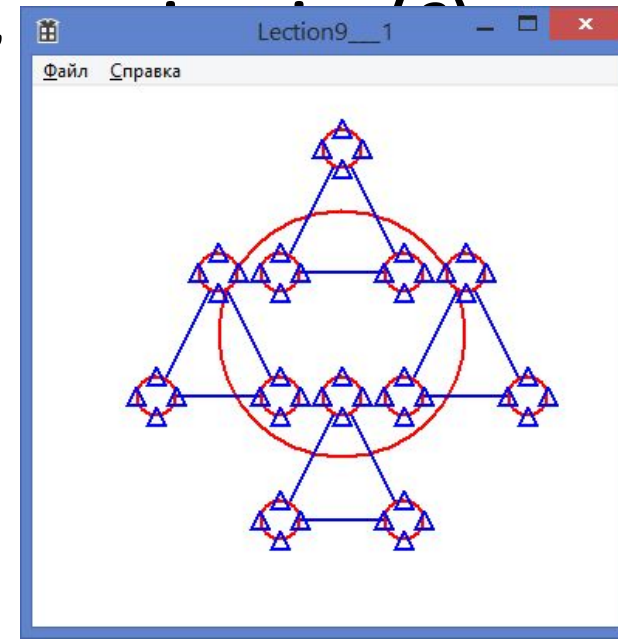


# Косвенная рекурсия (2)

```
void IndirectRecursiveTriangle(HDC hdc, int cx, int cy, int size)
{
    Triangle(hdc, cx, cy, size);
    if (size < 10) {
        return;
    }
    IndirectRecursiveCircle(hdc, cx, cy - size, size / 3);
    IndirectRecursiveCircle(hdc, cx + size, cy + size, size / 3);
    IndirectRecursiveCircle(hdc, cx - size, cy + size, size / 3);
}
```

...

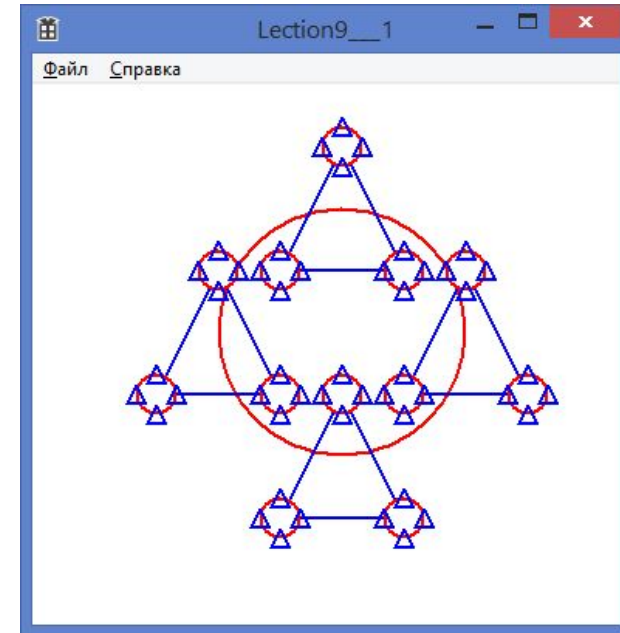
```
IndirectRecursiveCircle(hdc, 200, 160, 80);
```



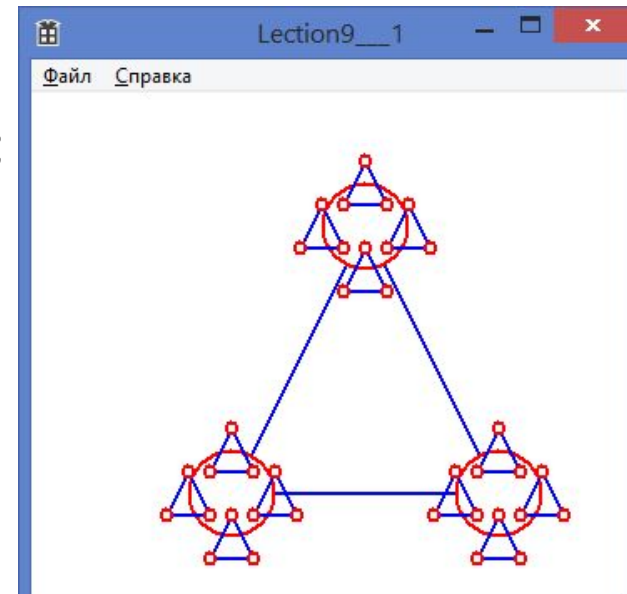


# Косвенная рекурсия (3)

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



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

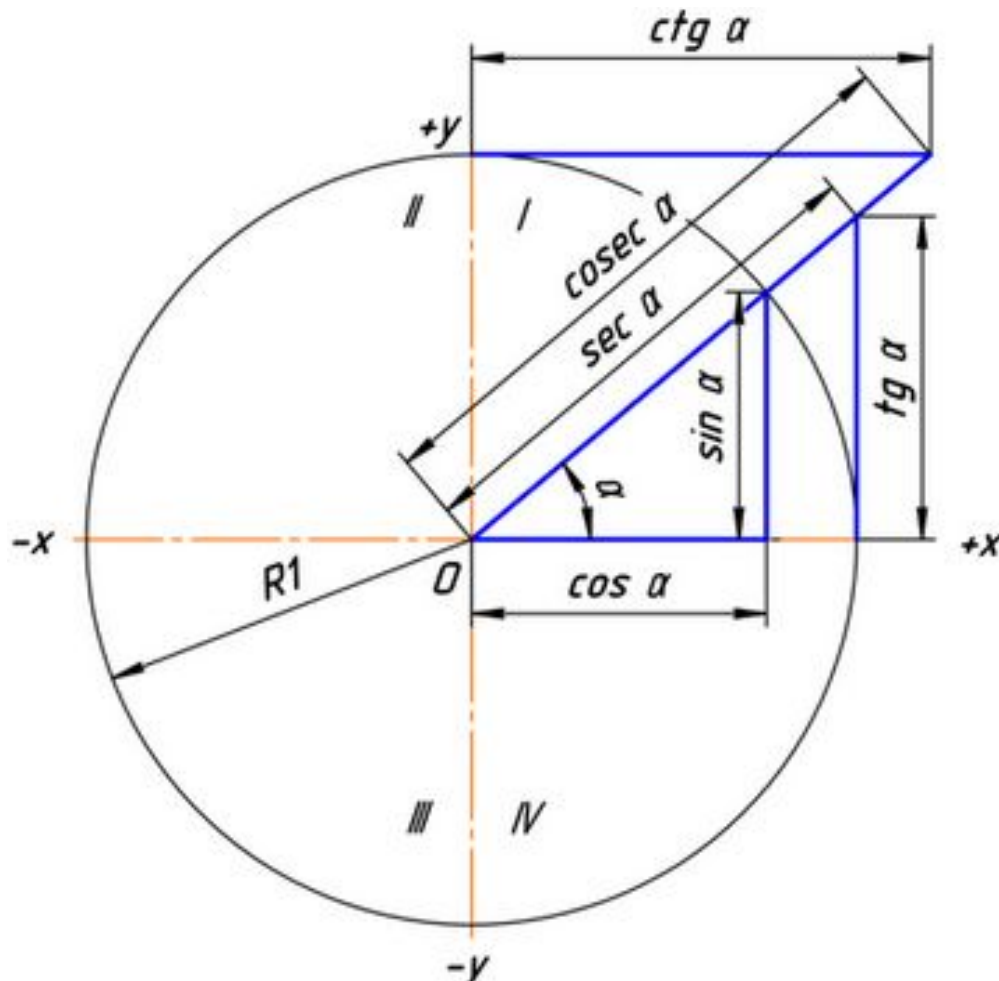


# Прототипы функций

```
void IndirectRecursiveCircle(HDC hdc, int cx, int cy, int size);
```

```
void IndirectRecursiveTriangle(HDC hdc, int cx, int cy, int size);
```

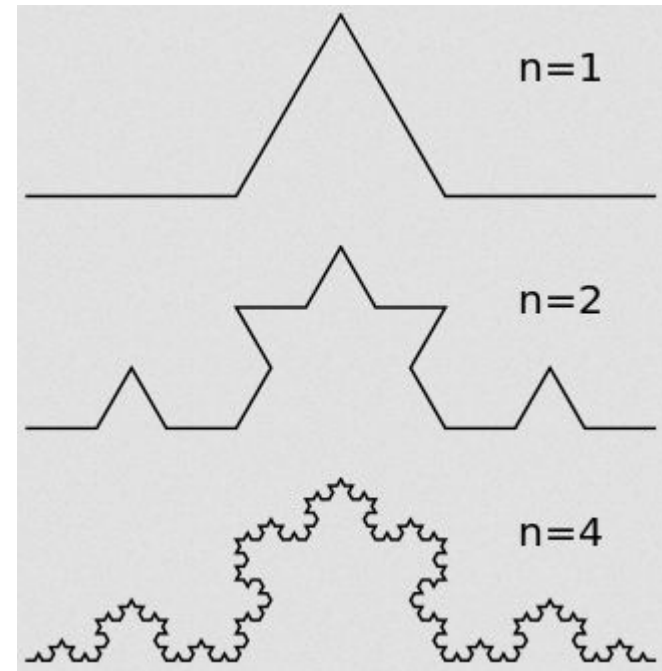
# И снова вспоминаем тригонометрию



Численные значения тригонометрических функций угла в тригонометрической окружности с радиусом, равным единице

# Фракталы. Кривая Коха

[https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%B2%D0%B0%D1%8F\\_%D0%9A%D0%BE%D1%85%D0%B0](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%B2%D0%B0%D1%8F_%D0%9A%D0%BE%D1%85%D0%B0)



# Фракталы. Кривая Коха

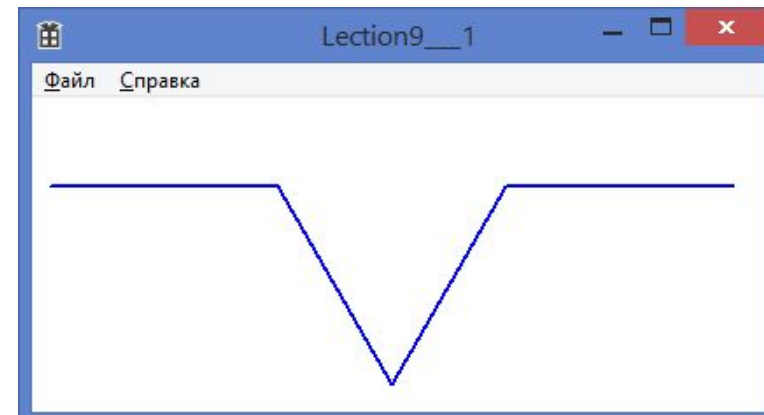
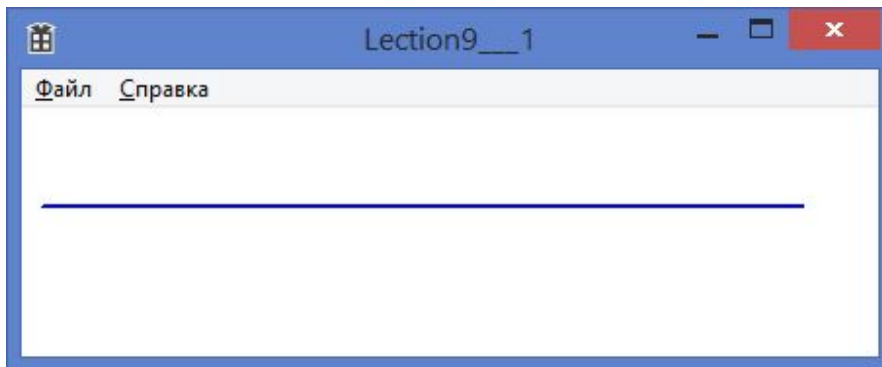
// <http://javatalks.ru/topics/11238> - ВЗЯТО ОТСЮДА

```
void drawKochLine(HDC hdc,  
    double ax, double ay,  
    double bx, double by,  
    double fi,  
    int n);
```

...

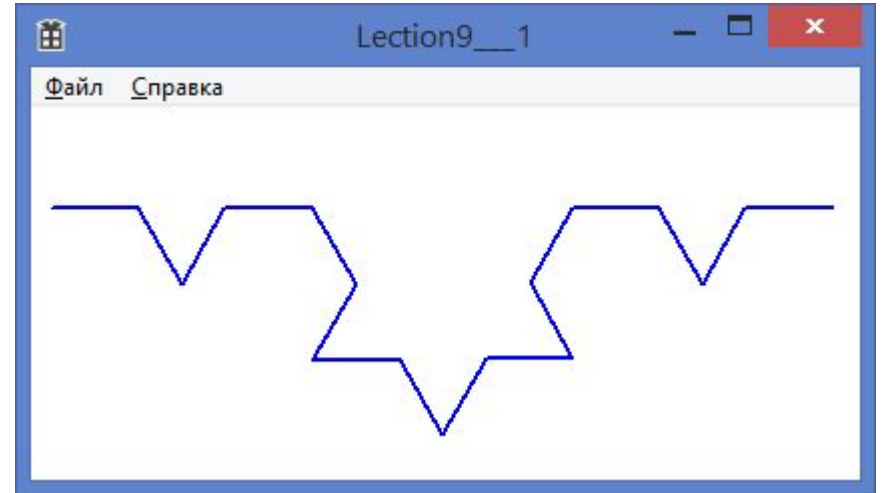
```
drawKochLine(hdc, 10, 50, 400, 50, 0, 0);
```

```
drawKochLine(hdc, 10, 50, 400, 50, 0, 1);
```

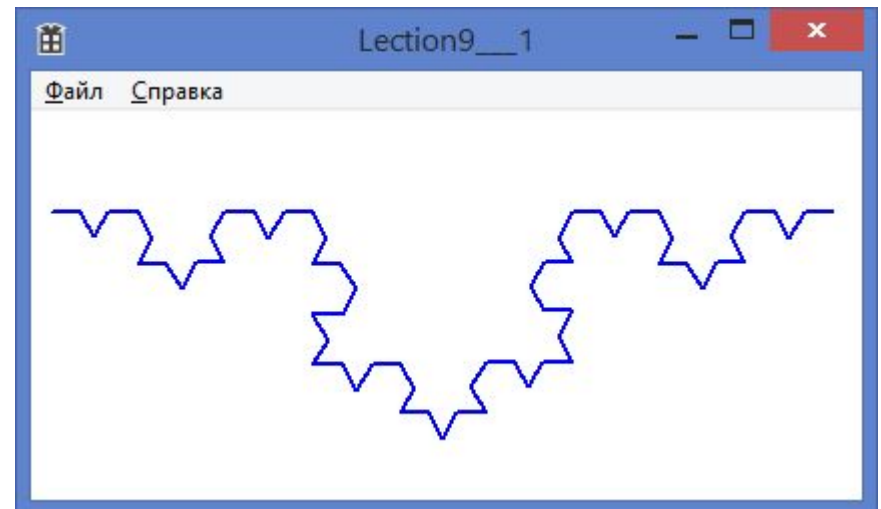


# Фракталы. Кривая Коха

```
drawKochLine(hdc, 10, 50, 400, 50, 0, 2);
```

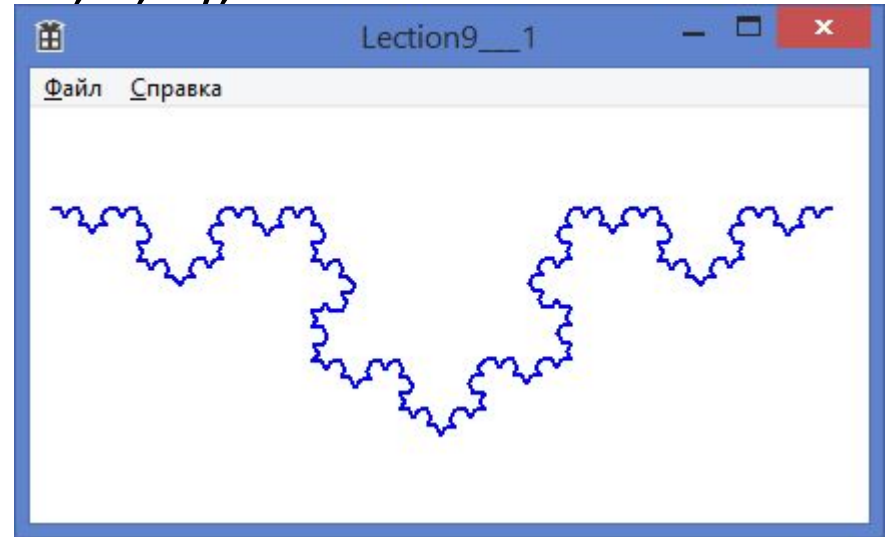


```
drawKochLine(hdc, 10, 50, 400, 50, 0, 3);
```

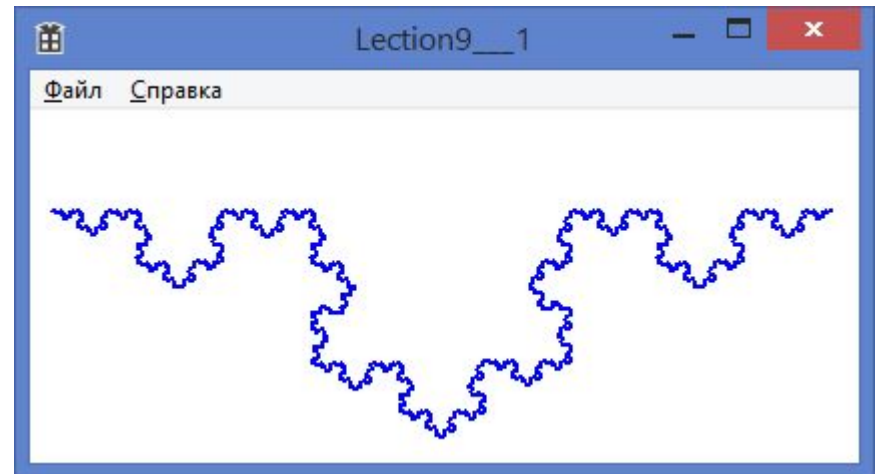


# Фракталы. Кривая Коха

```
drawKochLine(hdc, 10, 50, 400, 50, 0, 4);
```



```
drawKochLine(hdc, 10, 50, 400, 50, 0, 5);
```



# Функция рисующая кривую Коха

```
void drawKochLine(HDC hdc,  
    double ax, double ay,  
    double bx, double by,  
    double fi,  
    int n) {  
  
    if (n <= 0) {  
        // рисуем прямую, если достигнута необходимая  
        // глубина рекурсии.  
        HPEN hPen;  
        hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));  
        SelectObject(hdc, hPen);  
        MoveToEx(hdc, ax, ay, NULL);  
        LineTo(hdc, bx, by);  
        DeleteObject(hPen);  
    }  
}
```



# Функция рисующая кривую Коха

```
else {  
    // находим длину отрезка (a; b).  
    double dx = ax - bx;  
    double dy = ay - by;  
    double length = sqrt(dx * dx + dy * dy);  
  
    // находим длину 1/3 отрезка (a; b)  
    double length1of3 = length / 3;  
  
    // находим точку делящую отрезок как 1:3.  
    double a1x = ax + round((length1of3 * cos(fi)));  
    double a1y = ay + round((length1of3 * sin(fi)));  
  
    // находим точку делящую отрезок как 2:3.  
    double b1x = a1x + round((length1of3 * cos(fi)));  
    double b1y = a1y + round((length1of3 * sin(fi)));
```

# Функция рисующая кривую Коха

```
// находим точку, которая будет вершиной
// треугольника.
double cx = a1x + round((length1of3 * cos(fi + M_PI / 3)));
double cy = a1y + round((length1of3 * sin(fi + M_PI / 3)));

drawKochLine(hdc, a1x, a1y, cx, cy, fi + M_PI / 3, n - 1);
drawKochLine(hdc, cx, cy, b1x, b1y, fi - M_PI / 3, n - 1);

drawKochLine(hdc, ax, ay, a1x, a1y, fi, n - 1);
drawKochLine(hdc, b1x, b1y, bx, by, fi, n - 1);
} // конец else
} // конец функции
```

# Домашнее задание

1. Воспроизвести все рекурсивные (прямые) функции отрисовки (крест, треугольник, круг).
2. Поэкспериментировать с рекурсией – создать свой собственный рисунок из рекурсивно повторяющихся фигур
3. \*\* Воспроизвести косвенную рекурсию. Поэкспериментировать с ними
4. \*\*\* Воспроизвести функцию отрисовки кривой Коха
5. +++ Нарисовать снежинку Коха или любой другой красивый фрактал

# Источники информации

- **КАК рисовать в Win32 API?**

-<http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html>

- **Документация по Win API – MSDN**  
(Лучше искать через Google)

- **Фракталы**

-<https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%B%D>  
[В](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%B%D)