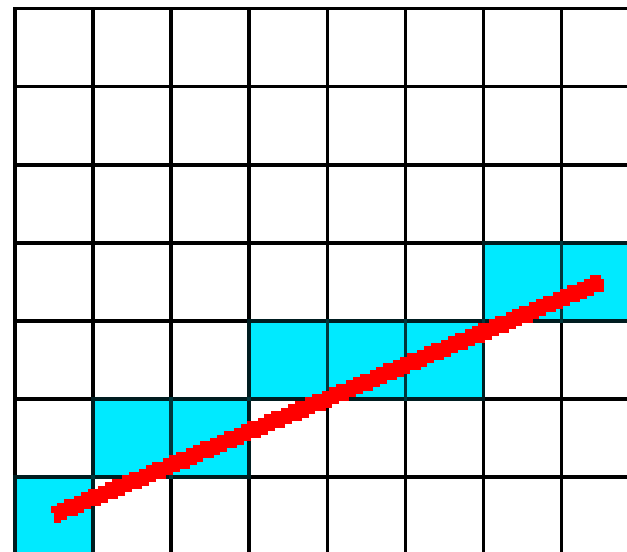
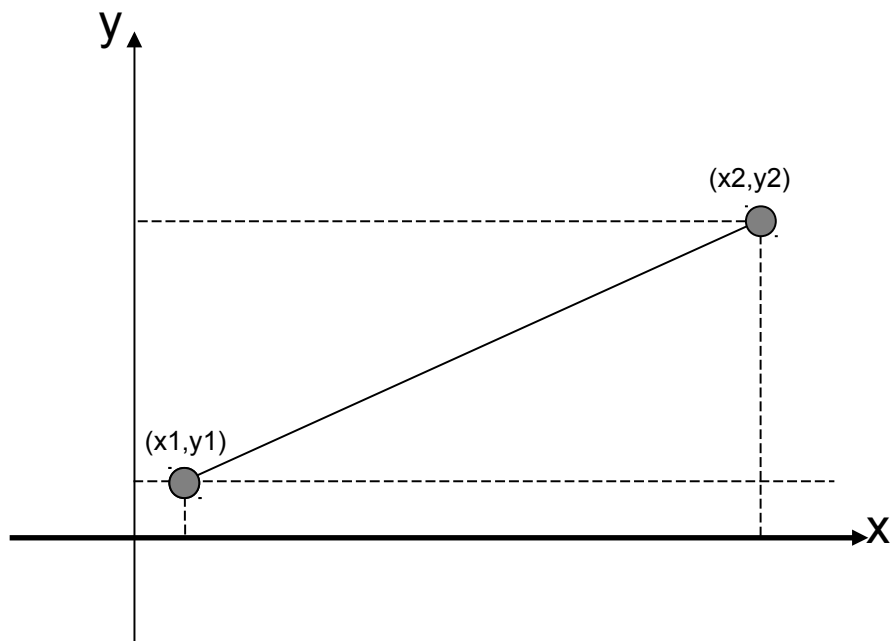


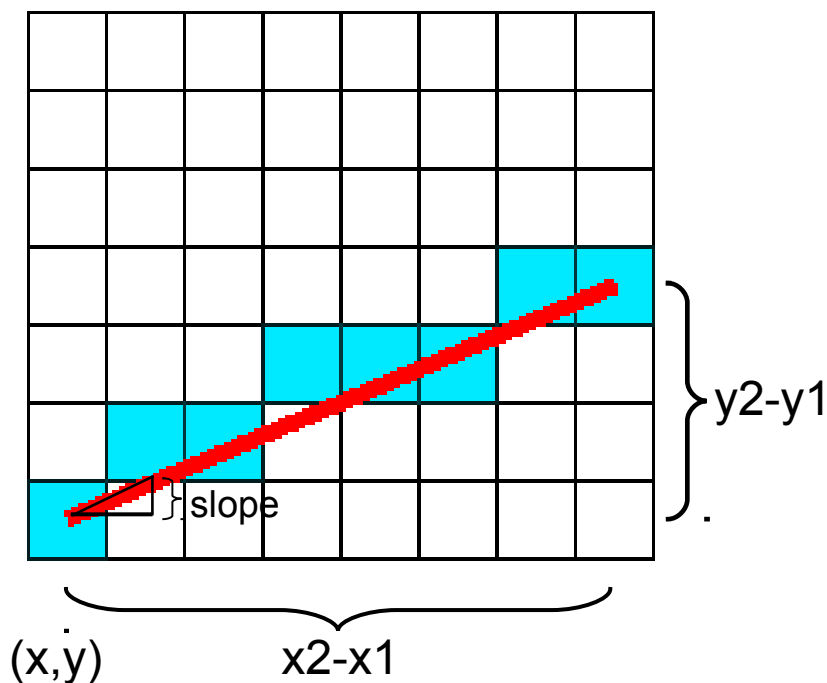
# Растровая графика

URL: <http://www.school30.spb.ru/cgsg/cgc/>

E-mail: [CGSG@yandex.ru](mailto:CGSG@yandex.ru)

- Точки
- Линии
- Прямоугольники (со сторонами, параллельными границам экрана)
- Многоугольники
- Шрифты
- Заливка областей
- Плоское отсечение

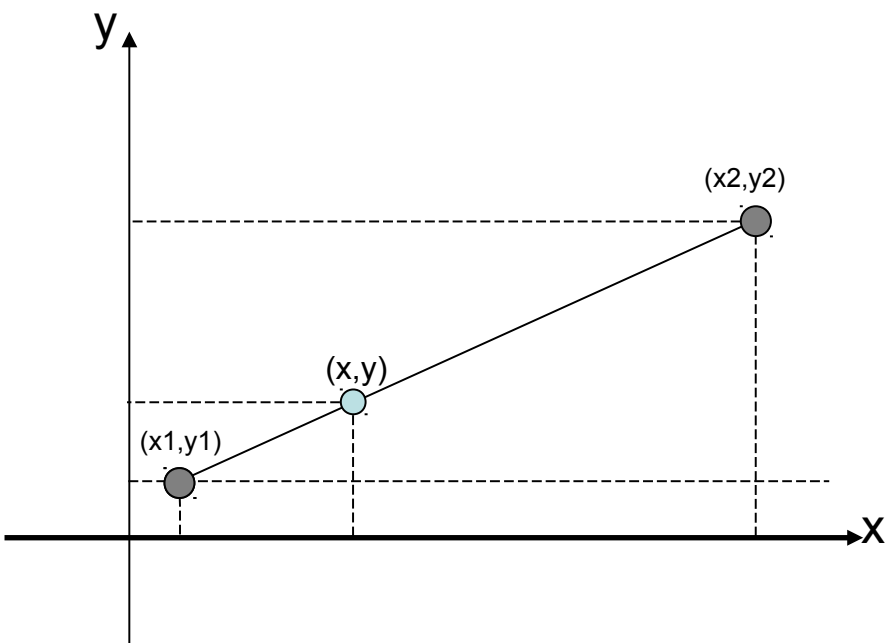




```
int x;
float
    y,
    slope = (y2 - y1) / (x2 - x1);

x = x1; y = y1 + 0.5;

while (x <= x2)
{
    SetPixel(x, (int)y);
    y = y + slope;
    x = x + 1;
}
```



$$\frac{x - x1}{x2 - x1} = \frac{y - y1}{y2 - y1}$$

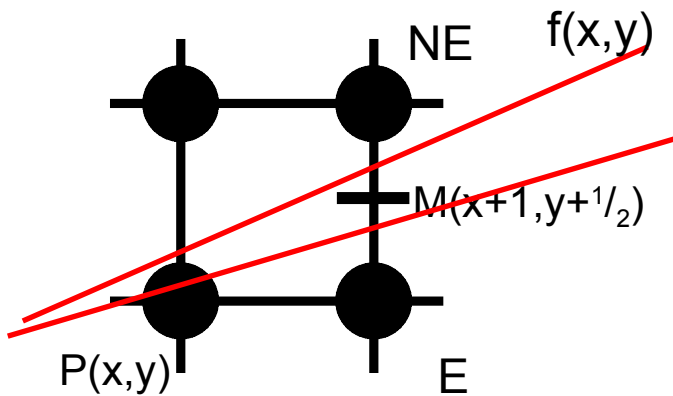
$$(x - x1) \cdot (y2 - y1) - (y - y1) \cdot (x2 - x1) = 0$$

$$dy \cdot x - dx \cdot y - (x1 \cdot dy - y1 \cdot dx) = 0$$

$$f(x, y) = dy \cdot x - dx \cdot y - (x1 \cdot dy - y1 \cdot dx)$$

$$\begin{cases} f(x, y) > 0 & \text{точка } (x, y) \text{ «ниже» прямой} \\ f(x, y) = 0 & \text{точка } (x, y) \text{ «лежит» на прямой} \\ f(x, y) < 0 & \text{точка } (x, y) \text{ «выше» прямой} \end{cases}$$

# Line: Алгоритм Брезенхема (метод центральной точки)



Подставляем точку M в функцию  $f$ :

- если  $f(M) > 0$  выбираем точку NE
- если  $f(M) \leq 0$  выбираем точку E

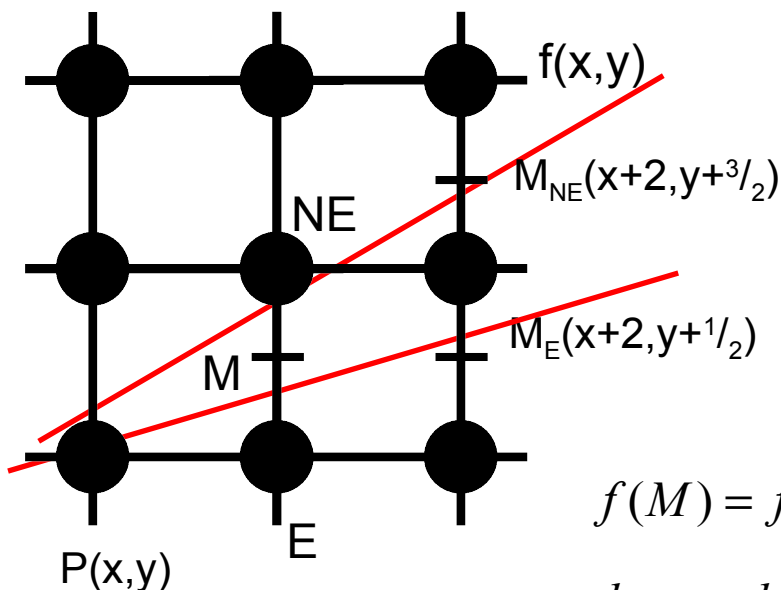
```
int x, y;

x = x1; y = y1;

SetPixel(x, y);
count = dx;
while (count > 0)
{
    count = count - 1;

    if (f(x + 1, y + 0.5) > 0)
        y = y + 1;
    x = x + 1;
    SetPixel(x, y);
}

/* f(x, y) =
 *      dy * x - dx * y - (x1 * dy - y1 * dx)
 * dx = x2 - x1; dy = y2 - y1;
 */
```



Подставляем точку M в функцию f:

- если  $f(M) > 0$  выбираем точку NE
- если  $f(M) \leq 0$  выбираем точку E

Изменения значения  $f(M)$  при переходе к новым точкам (E или NE):

$$f(M) = f(x+1, y + \frac{1}{2}) = dy \cdot (x+1) - dx \cdot (y + \frac{1}{2}) - C = dy \cdot x + dy - dx \cdot y - \frac{dx}{2} - C$$

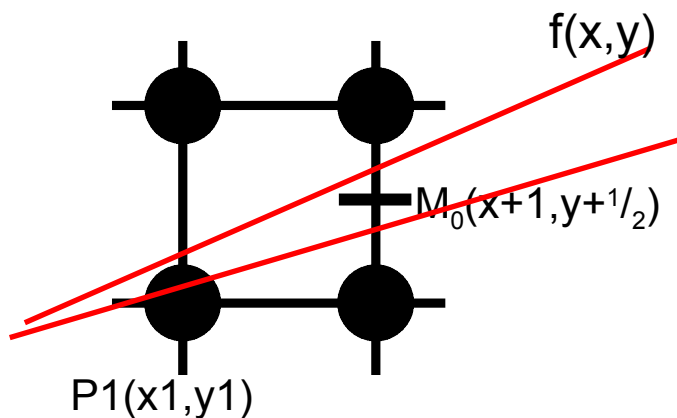
$$f(M_E) = f(x+2, y + \frac{1}{2}) = dy \cdot (x+2) - dx \cdot (y + \frac{1}{2}) - C = dy \cdot x + 2 \cdot dy - dx \cdot y - \frac{dx}{2} - C = \underline{f(M) + dy}$$

$$f(M_{NE}) = f(x+2, y + \frac{3}{2}) = dy \cdot (x+2) - dx \cdot (y + \frac{3}{2}) - C = dy \cdot x + 2 \cdot dy - dx \cdot y - 3 \cdot \frac{dx}{2} - C = \underline{f(M) + dy - dx}$$

# Line: Алгоритм Брезенхема (метод центральной точки)

Известны приращения  $f$ .

Найдем первоначальное значение для точки  $(x_1, y_1)$



$$f(M_0) = f(x_1 + 1, y_1 + \frac{1}{2}) =$$

$$dy \cdot (x_1 + 1) - dx \cdot (y_1 + \frac{1}{2}) - (x_1 \cdot dy - y_1 \cdot dx) =$$

$$dy - \frac{dx}{2}$$



Сохранились вещественные числа.

Сделаем замену:  $2f = e$

Тогда помеченные строки изменяться на:

$e = 2 * dy - dx;$

$e > 0$

$e = e + 2 * dy - 2 * dx;$

$e = e + 2 * dy$

и  $e$  – целое число.

```
int x, y, dx, dy;
float f;

dx = x2 - x1;
dy = y2 - y1;
f = dy - dx / 2.0;

x = x1; y = y1;
SetPixel(x, y);
count = dx;
while (count > 0)
{
    count = count - 1;

    if (f > 0)
    {
        y = y + 1;
        f = f + (dy - dx);
    }
    else
    {
        f = f + dy;
        x = x + 1;
        SetPixel(x, y);
    }
}
```

```
int x, y, dx, dy, incrE, incrNE, e;

dx = x2 - x1;
dy = y2 - y1;
e = 2 * dy - dx;
incrE = 2 * dy;
incrNE = 2 * dy - 2 * dx;

x = x1; y = y1;
SetPixel(x, y);
count = dx;
while (count > 0)
{
    count = count - 1;

    if (f > 0)
    {
        y = y + 1;
        f = f + incrNE;
    }
    else
        f = f + incrE;
    x = x + 1;
    SetPixel(x, y);
}
```

Fixed Point – вещественные числа с фиксированной точкой.

Рассмотрим 4-байтное целое:

2b целая часть	2b дробная часть
----------------	------------------

Точность  $1/_{65536}$

Если  $x$  и  $y$  fixed point, то

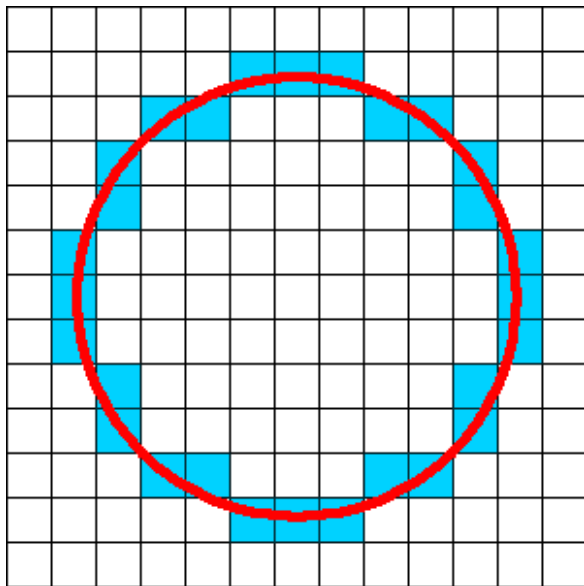
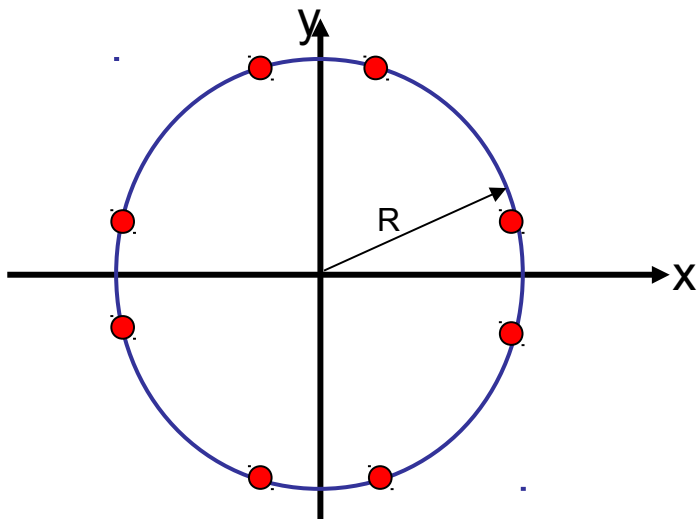
- сложение не изменяется ( $x+y$ )
- вычитание не изменяется ( $x-y$ )
- целая часть – «двоичный сдвиг» вправо на 16 бит ( $x \gg 16$ )
- из целого:  $x = a \ll 16$

```
int x;
long
y,
slope = ((y2 - y1) << 16) / (x2 - x1);

x = x1; y = y1 << 16;

SetPixel(x1, y1);
count = x2 - x1;
while (count > 0)
{
    count = count - 1;

    y = y + slope;
    x = x + 1;
    SetPixel(x, y >> 16);
}
```



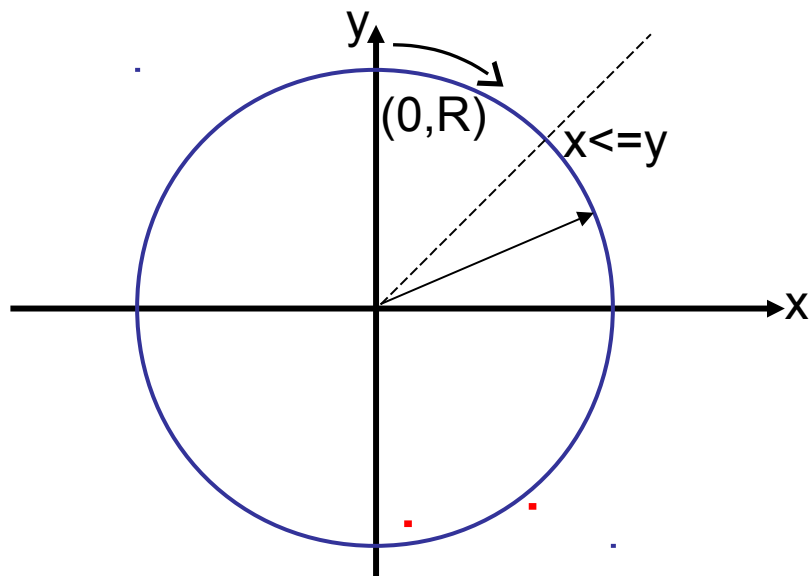
```
SetPixel4(x, y):
```

```
SetPixel(Cx, Cy + R);
SetPixel(Cx, Cy - R);
SetPixel(Cx + R, Cy);
SetPixel(Cx - R, Cy);
```

```
SetPixel8(x, y):
```

```
SetPixel(Cx + x, Cy + y);
SetPixel(Cx - x, Cy + y);
SetPixel(Cx + x, Cy - y);
SetPixel(Cx - x, Cy - y);
SetPixel(Cx + y, Cy + x);
SetPixel(Cx - y, Cy + x);
SetPixel(Cx + y, Cy - x);
SetPixel(Cx - y, Cy - x);
```

# Circle: Алгоритм Брезенхема (метод центральной точки)



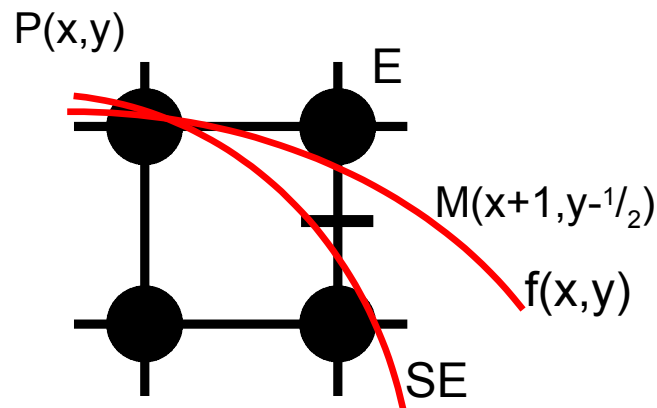
```
int x, y;

x = 0; y = R;

SetPixel4(x, y);
while (x <= y)
{
    if (f(x + 1, y - 0.5) < 0)
        y = y - 1;
    x = x + 1;
    SetPixel8(x, y);
}
```

$$f(x, y) = x^2 + y^2 - R^2$$

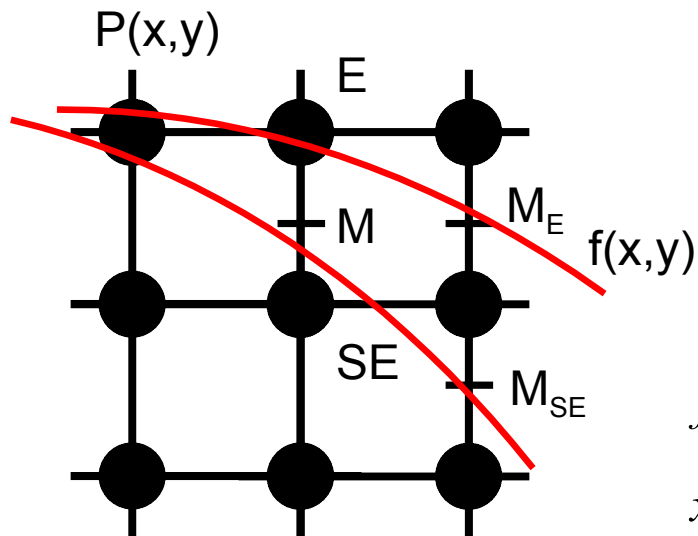
$$\begin{cases} f(x, y) > 0 & \text{точка } (x, y) \text{ вне круга} \\ f(x, y) = 0 & \text{точка } (x, y) \text{ на окружности} \\ f(x, y) < 0 & \text{точка } (x, y) \text{ внутри круга} \end{cases}$$



Подставляем точку M в функцию f:

- если  $f(M) \geq 0$  выбираем точку SE
- если  $f(M) < 0$  выбираем точку E

# Circle: Алгоритм Брезенхема (метод центральной точки)



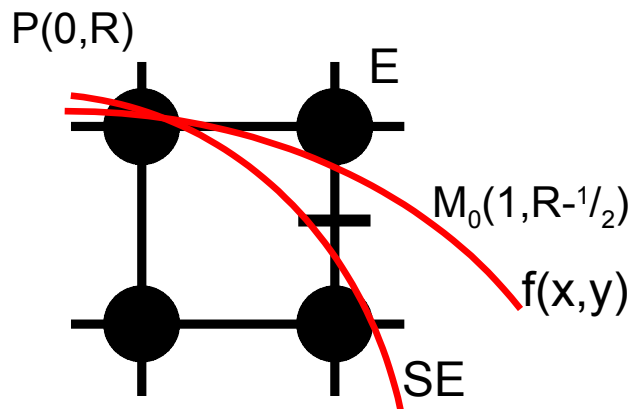
Изменения значения  $f(M)$  при переходе к новым точкам (E или SE):

$$f(M) = f(x+1, y - \frac{1}{2}) = (x+1)^2 + (y - \frac{1}{2})^2 - R^2 = x^2 + 2x + 1 + y^2 - y + \frac{1}{4} - R^2$$

$$f(M_E) = f(x+2, y - \frac{1}{2}) = (x+2)^2 + (y - \frac{1}{2})^2 - R^2 = x^2 + 4x + 4 + y^2 - y + \frac{1}{4} - R^2 = f(M) + 2x + 3$$

$$f(M_{NE}) = f(x+2, y - \frac{3}{2}) = (x+2)^2 + (y - \frac{3}{2})^2 - R^2 = x^2 + 4x + 4 + y^2 - 3y + \frac{9}{4} - R^2 = f(M) + 2x - 2y + 5$$

# Circle: Алгоритм Брезенхема (метод центральной точки)



Определили приращения f.

Найдем первоначальное значение для точки (x1, y1)

$$f(M_0) = f(1, R - \frac{1}{2}) = 1^2 + (R - \frac{1}{2})^2 - R^2 =$$

$$1 + R^2 - R + \frac{1}{4} - R^2 = \frac{5}{4} - R$$

```
int x, y, f = 1 - R;

x = 0; y = R;

SetPixel4(x, y);
while (x <= y)
{
    if (f > 0)
    {
        y = y - 1;
        f = f + 2 * (x - y) + 5;
    }
    else
    {
        f = f + 2 * x + 3;
        x = x + 1;
        SetPixel8(x, y);
    }
}
```

Все приращения - целые. Сравнение f с 0 строгое: '<'.

Поэтому из первоначального f можно вычесть  $\frac{1}{4}$ .

Дополнительная оптимизация:

Просчитаем изменение приращений по направлениям E и SE ( $incrE=2*x+3$  и  $incrSE=2*(x-y)+5$ ) для избавления от доступа к переменным.

- Если выбрана точка E, то 'x' увеличивается на 1 и:

$$\underline{incrE=incrE+2} \text{ и } \underline{incrSE=incrSE+2}$$

- Если выбрана точка SE, то 'x' увеличивается на 1, 'y' уменьшается на 1 и:

$$\underline{incrE=incrE+2} \text{ и } \underline{incrSE=incrSE+4}$$

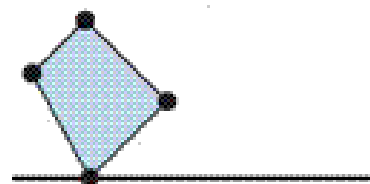
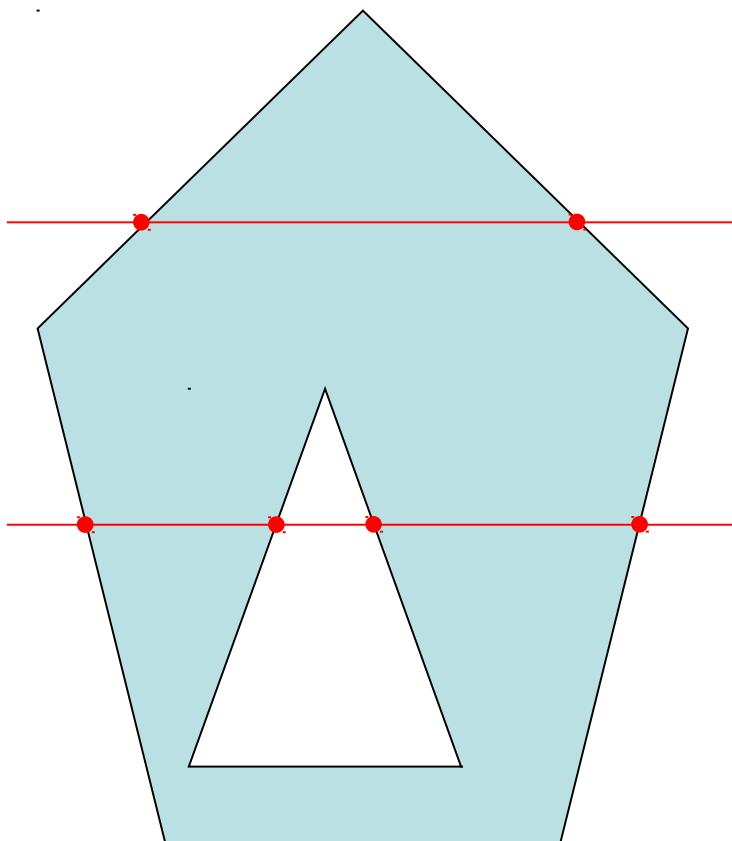
Изначальные значения:

$$\underline{incrE=3} \text{ и } \underline{incrSE=5-2*R}$$

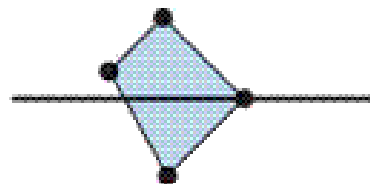


```
int
    x = 0, y = R,
    f = 1 - R,
    incrE = 3,
    incrSE = 5 - 2 * R;

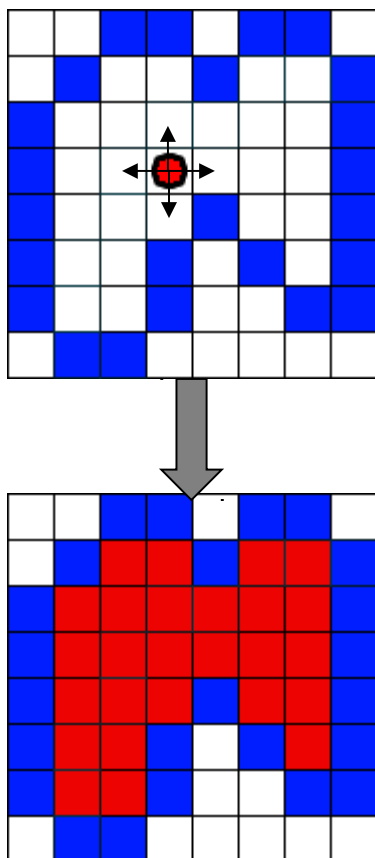
SetPixel4(x, y);
while (x <= y)
{
    if (f > 0)
    {
        y = y - 1;
        f = f + incrSE;
        incrSE = incrSE + 4;
    }
    else
    {
        f = f + incrE;
        incrSE = incrSE + 2;
    }
    incrE = incrE + 2;
    x = x + 1;
    SetPixel8(x, y);
}
```



(a)



(b)



`Push(x, y)` – сохранить на стеке  
`Pop(&x, &y)` – получить со стека  
`Get(x, y)` – получить цвет точки  
`Color` – цвет покраски

```

Back = Get(x, y);
if (Back == Color)
    return;
Push(x, y);
while (стек не пуст)
{
    Pop(&x, &y);
    if (точка(x, y) на экране &&
        Back == Color)
    {
        SetPixel(x, y, Color);
        Push(x + 1, y);
        Push(x - 1, y);
        Push(x, y + 1);
        Push(x, y - 1);
    }
}
    
```

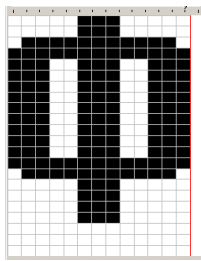
```

Back = Get(x, y);
if (Back == Color)
    return;
Push(x, y);
while (стек не пуст)
{
    Pop(&x, &y);
    /* Поиск границ */
    left = x - 1;
    while (left в экране && Get(left, y) == Back)
        left--;
    left++;
    right = x + 1;
    while (right в экране && Get(right, y) == Back)
        right++;
    right--;
    /* Отрисовка уровня */
    Line(left, y, right, y, Color);
    /* Поиск смежных областей */
    pos = left; y = y - 1;
    while (pos <= right)
    {
        /* пропускаем поочередно зоны покраски/непокраски */
        while (pos <= right && Get(pos, y) != Back)
            pos++;
        if (Get(pos, y) == Back)
        {
            Push(pos, y);
            while (pos <= right && Get(pos, y) == Back)
                pos++;
        }
    }
    /* аналогично с уровнем y + 2 */
    . . .
}

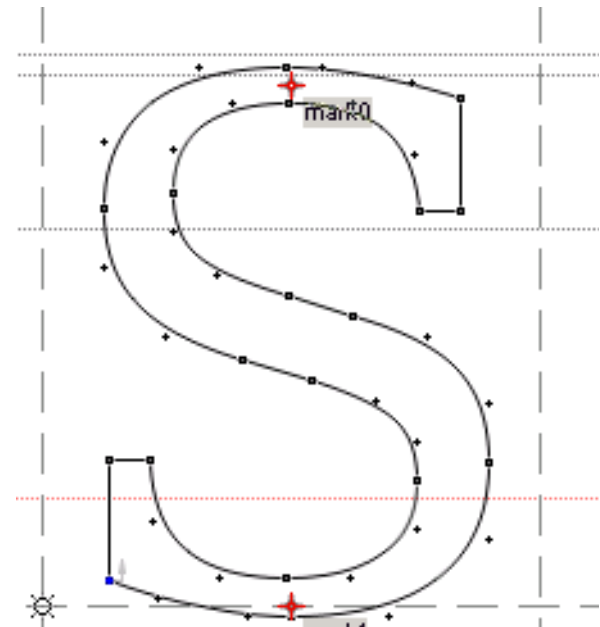
```

Шрифты		
Растровые	Векторные	Контурные

!	"	Э	Ч	%	'	(	)
*	+	-	/	0	1	2	3
4	5	6	7	8	9	:	<
>	?	а	б	в	г	д	е
ж	з	и	к	л	м	н	о
п	р	с	т	у	ф	х	ц
ш	щ	ъ	ы	я	з	и	й
!	"	Э	Ч	%	'	(	)
*	+	-	/	0	1	2	3
4	5	6	7	8	9	:	<
>	?	а	б	в	г	д	е
ж	з	и	к	л	м	н	о
п	р	с	т	у	ф	х	ц
ш	щ	ъ	ы	я	з	и	й
!	"	Э	Ч	%	'	(	)
*	+	-	/	0	1	2	3
4	5	6	7	8	9	:	<
>	?	а	б	в	г	д	е
ж	з	и	к	л	м	н	о
п	р	с	т	у	ф	х	ц
ш	щ	ъ	ы	я	з	и	й



AaBbCc  
*Script*



Аа Aa Aa

								0x3C
								0x46
								0x86
								0x86
								0x86
								0x86
								0xFE
								0x86
								0x00

Справа показана битовая кодировка  
каждой строки  
(в шестнадцатеричном виде)

```
unsigned char *Table = ...;
int i, j;

/* H - высота символов в строках */
Table += КодСимвола * H;
for (i = 0; i < H; i++)
{
    for (j = 0; j < 8; j++)
        if ((Table & (0x80 >> j)) != 0)
            SetPixel(X + j, Y + i);
    Table++;
}
```

- Упражнение

- Рекомендуется реализовать растровые алгоритмы с помощью программы из первого упражнения. Растровые шрифты необходимо загружать из файла.