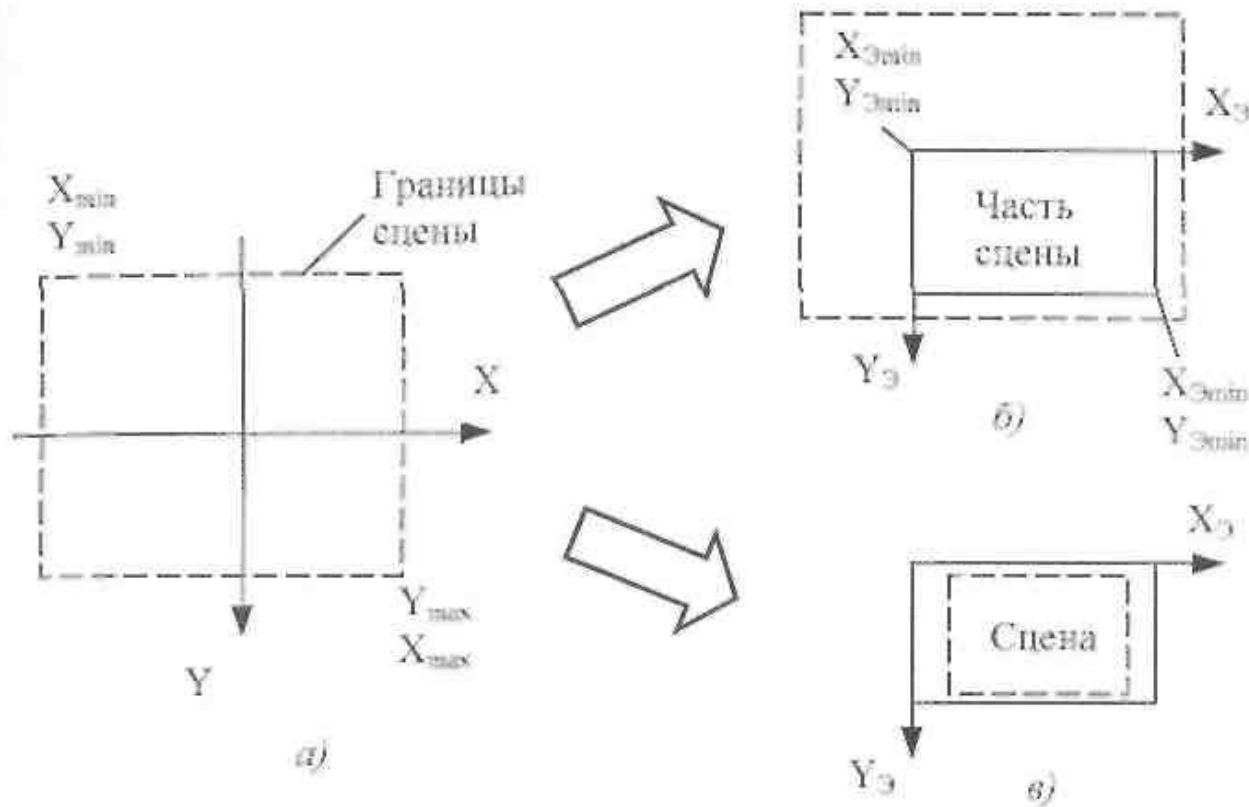


# Отображение в окне



. Отображение проекции сцены:  
а — границы сцены в координатах проекции; б — в окне часть сцены,  
в — вся сцена с сохранением пропорций вписана в окно

$$\begin{cases} X_3 = KX + dx, \\ Y_3 = KY + dy, \\ Z_3 = KZ. \end{cases} \quad \begin{cases} X_3 \min \leq KX_{\min} + dx, & (1) \\ Y_3 \min \leq KY_{\min} + dy, & (2) \\ X_3 \max \geq KX_{\max} + dx, & (3) \\ Y_3 \max \geq KY_{\max} + dy. & (4) \end{cases}$$

$$K \leq \frac{X_3 \max - X_3 \min}{X_{\max} - X_{\min}} = K_x.$$

$$K \leq \frac{Y_3 \max - Y_3 \min}{Y_{\max} - Y_{\min}} = K_y.$$

$$K \leq \min \{K_x, K_y\} = K_{\min}, \quad dx \geq X_3 \min - KX_{\min} = dx_1.$$

$$dx \leq X_3 \max - KX_{\max} = dx_2.$$

$$dx = \frac{dx_1 + dx_2}{2} = \frac{X_3 \min - KX_{\min} + X_3 \max - KX_{\max}}{2}.$$

$$dy = \frac{Y_3 \min - KY_{\min} + Y_3 \max - KY_{\max}}{2}.$$

# Выводы



$$\begin{pmatrix} X_3 \\ Y_3 \\ Z_3 \\ 1 \end{pmatrix} = \begin{pmatrix} K, dx, dy \end{pmatrix} \begin{pmatrix} \text{Повороты} \\ \alpha, \beta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} M_A \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

$$\begin{pmatrix} X_3 \\ Y_3 \\ Z_3 \\ 1 \end{pmatrix} = \begin{pmatrix} K, dx, dy \end{pmatrix} \begin{pmatrix} \text{Проециро-} \\ \text{вание} \end{pmatrix} \begin{pmatrix} \text{Повороты} \\ \alpha, \beta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

# Базовые растровые алгоритмы

## Алгоритмы вывода прямой линии

заданы координаты  $(x1, y1 - x2, y2)$  концов отрезка прямой

```
for (x=x1; x<=x2;
```

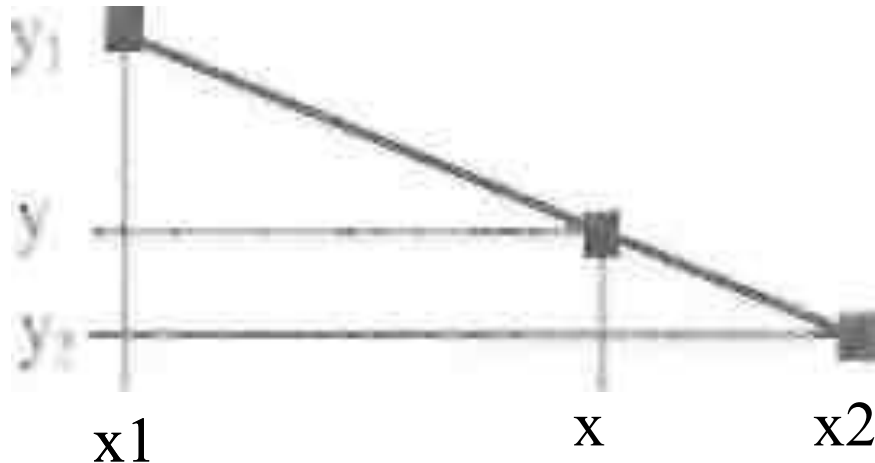
```
//Пиксел(x,
```

```
for (y=y1; y<=y2;
```

```
y++)
```

```
//Пиксел(x1, y);
```

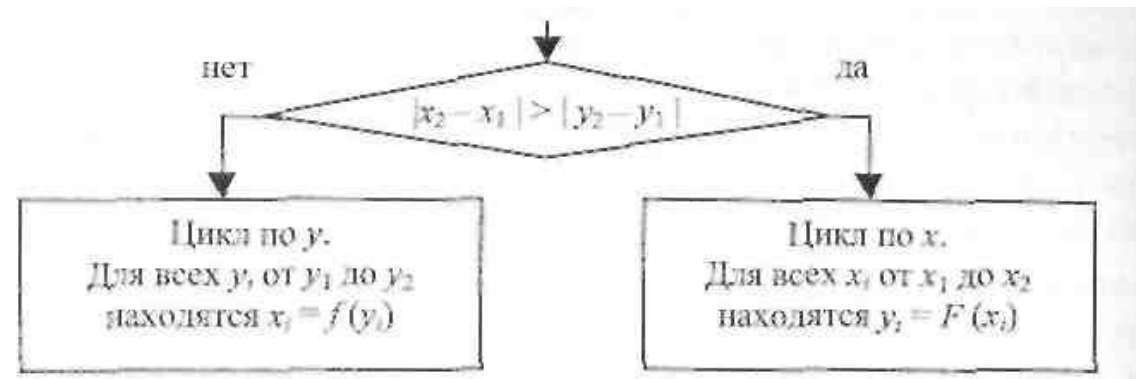
Прямое вычисление координат



$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1}$$

$$x = x_1 + (y - y_1) \frac{x_2 - x_1}{y_2 - y_1},$$

$$y = y_1 + (x - x_1) \frac{y_2 - y_1}{x_2 - x_1},$$



```

for (x=x1; x<=x2; x++)
{
    y=y1+((x-x1)*(y2-y1))/(x2-x1);
    // Пиксел(x,y);
}

```

```
float k;
```

```

k=(float)(y2-y1)/(float)(x2-x1);
for (x=x1; x<=x2; x++)
{
    y=y1+(float)(x-x1)*k;
    //Пиксел(x,y);
}

```

```

float yy,k;
k =(float)(y2-y1)/(float)(x2-x1);
yy=(float)y1 - (float)x1*k;
for (x=x1; x<=x2; x++)
{
    y = yy + (float)x*k;
    //Пиксел(x,y);
}

```

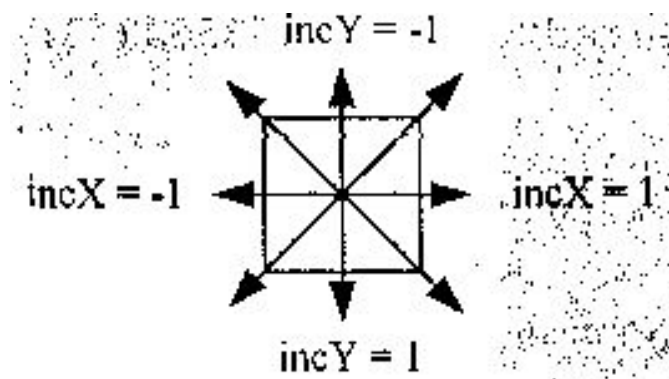
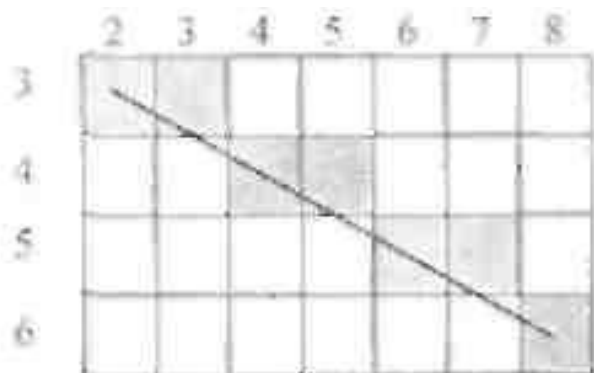
# Инкрементные алгоритмы

## Алгоритмы Брезенхема

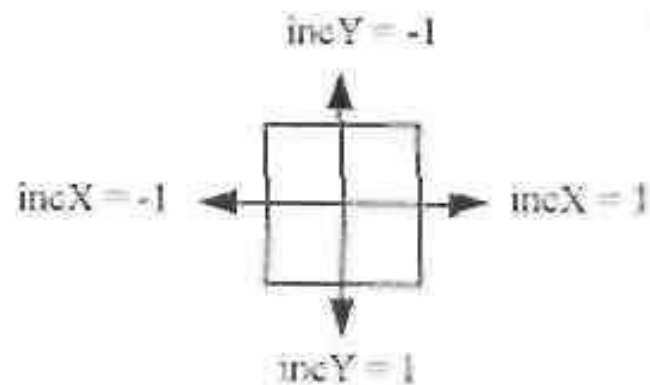
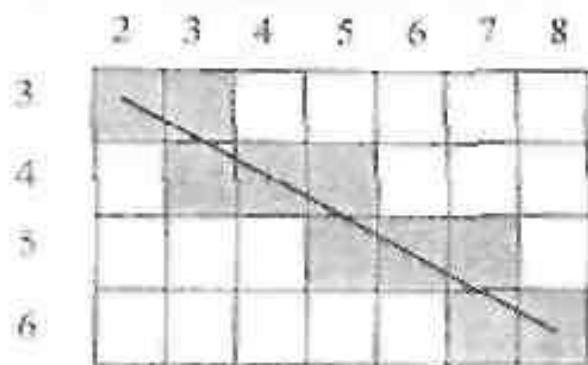
$xerr = 0, yerr = 0;$

$dx = x2 - x1, dy = y2 - y1;$

```
Если  $dx > 0$ , то  $incX = 1;$   
     $dx = 0$ , то  $incX = 0;$   
     $dx < 0$ , то  $incX = -1;$   
Если  $dy > 0$ , то  $incY = 1;$   
     $dy = 0$ , то  $incY = 0;$   
     $dy < 0$ , то  $incY = -1;$   
 $dx = |dx|, dy = |dy|;$   
Если  $dx > dy$ , то  $d = dx;$   
    иначе  $d = dy;$   
 $x = x1, y = y1;$   
Повторять (x, y)  
пока не достигнут  $d$  раз:  
{  
     $xerr = xerr + dx;$   
     $yerr = yerr + dy;$   
    Если  $xerr > d$ , то  $xerr = xerr - d$   
         $x = x + incX$   
    Если  $yerr > d$ , то  $yerr = yerr - d$   
         $y = y + incY$   
Повторять (x, y)
```



Восьмисвязность

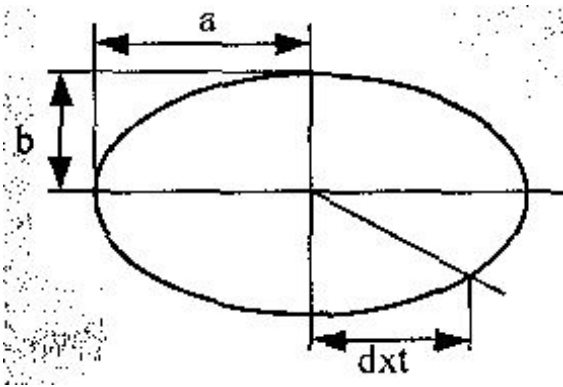


Четырехсвязность

## Алгоритм вывода окружности

$$X^2 + Y^2 = R^2$$

## Алгоритм вывода эллипса



$$dxt = \frac{a^2}{\sqrt{a^2 + b^2}},$$

## Кривая Безье

Кривые Безье описываются в параметрической форме:

$$\begin{aligned}x &= P_x(t), \\ y &= P_y(t)\end{aligned}$$



Многочлены Безье для  $P_x$  и  $P_y$

$$P_x(t) = \sum_{i=0}^m C_m^i t^i (1-t)^{m-i} x_i,$$

$$P_y(t) = \sum_{i=0}^m C_m^i t^i (1-t)^{m-i} y_i.$$

$C_m^i = m! / (i! (m-i)!)$   
сочетание  $m$  по  $i$

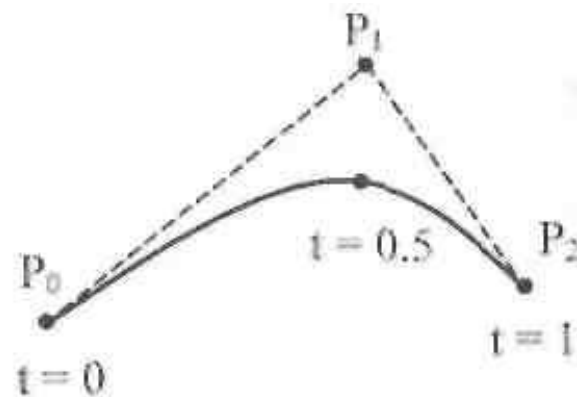
$m = 1$  (по двум точкам)

$$P(t) = (1-t) P_0 + t P_1.$$



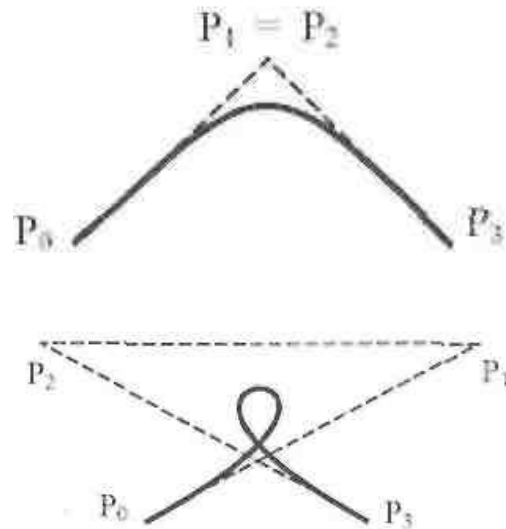
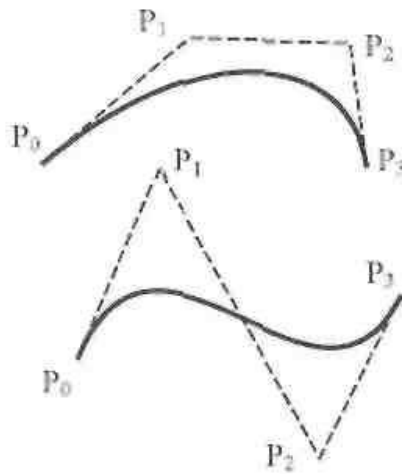
$m = 2$  (по трем точкам)

$$P(t) = (1-t)^2 P_0 + 2t(1-t) P_1 + t^2 P_2.$$

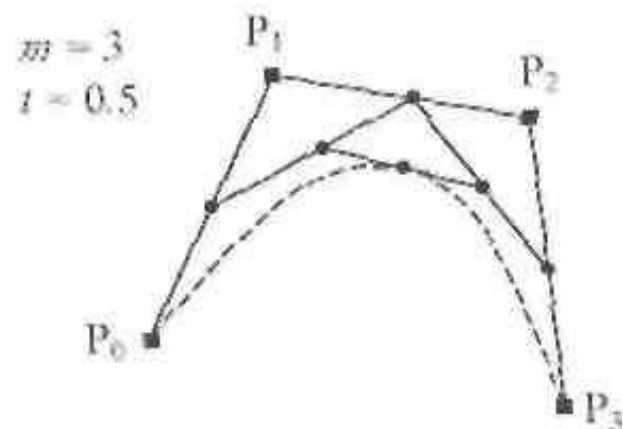
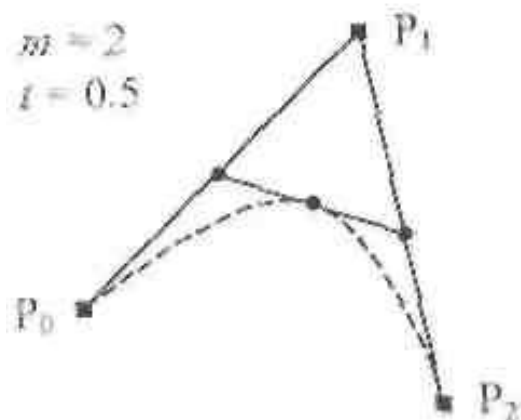


$m = 3$  (по четырем точкам, кубическая)

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t(1-t)^2 P_2 + t^3 P_3.$$

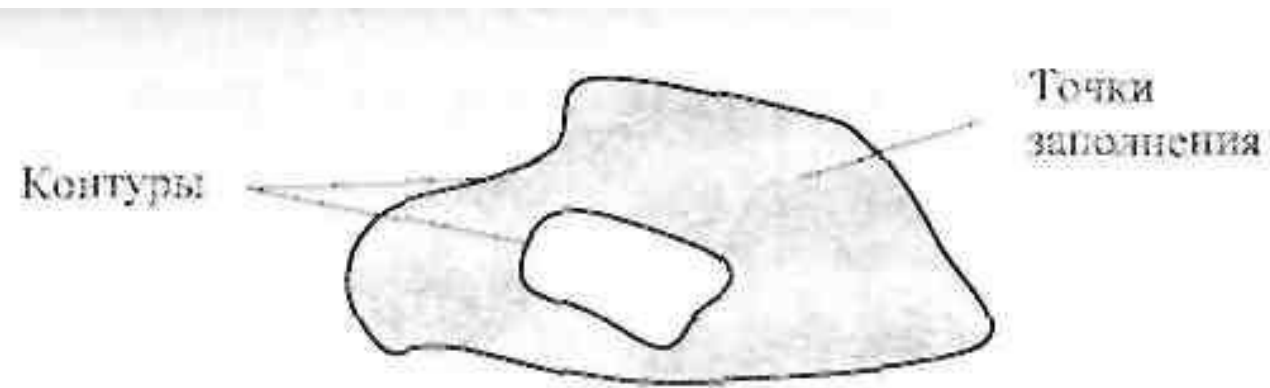


## Геометрический алгоритм для кривой Безье



```
for ( i = 0; i <= m; i++)  
    R[i] = P[i]; //формируем вспомогательный массив R[ ]  
for ( j = m; j > 0; j-- )  
    for ( i = 0; i < j; i++)  
        R[i] = R[i] + t * (R[i+1] - R[i]);
```

## Алгоритмы вывода фигур



## Алгоритмы закрашивания



```

функция ЗАКРАШИВАНИЕ(x, y)
{
    Если цвет пиксела (x, y) не равен цвету границы, то
    {
        установить для пиксела (x, y) цвет заполнения;
        ЗАКРАШИВАНИЕ(x+1, y);
        ЗАКРАШИВАНИЕ(x-1, y);
        ЗАКРАШИВАНИЕ(x, y+1);
        ЗАКРАШИВАНИЕ(x, y-1);
    }
}

```

## Волновой алгоритм закрашивания



3



39



67



179



159

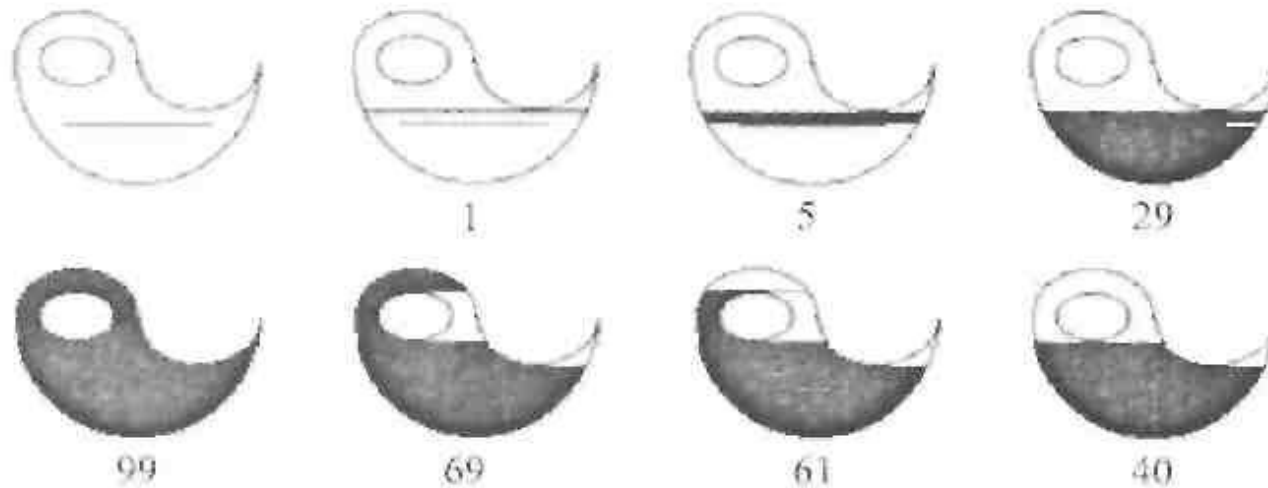


135



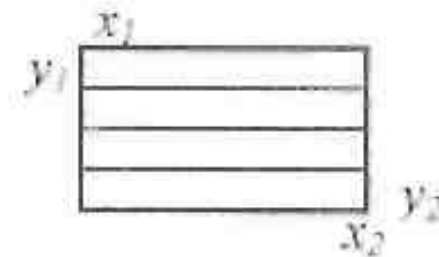
99

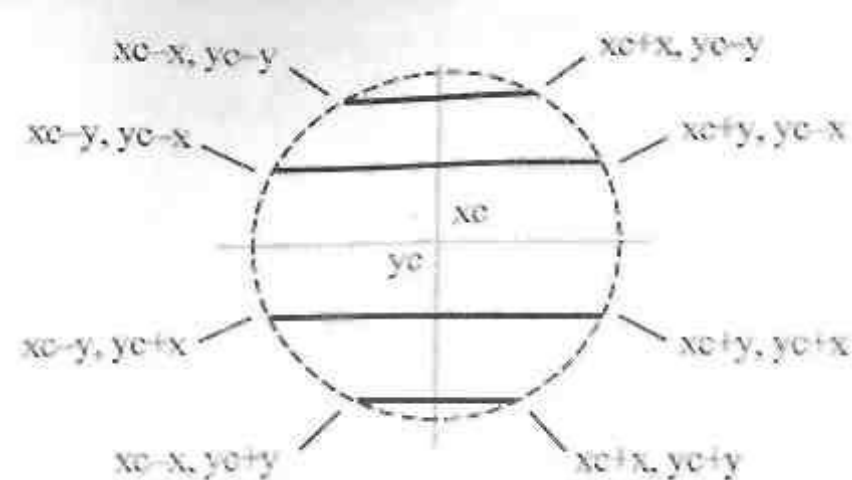
## Алгоритм закрашивания линиями



## Алгоритмы заполнения, которые используют математическое описание контура

```
for (y=y1; y<=y2; y++);  
//Рисуем горизонтальную линию  
//с координатами (x1, y) - (x2, y)
```





## Алгоритм ХУ

1. Найти  $\min\{y_i\}$  и  $\max\{y_i\}$  среди всех вершин  $P_i$ .
2. Выполнить цикл по  $y$  от  $y = \min$  до  $y = \max$ 
  - 3. { Нахождение точек пересечения всех отрезков контура с горизонталью  $y$ . Координаты  $X_i$  точек сечения записать в массив.
  - 4. Сортировка массива  $\{x_i\}$  по возрастанию  $x$ .
  - 5. Вывод горизонтальных отрезков с координатами

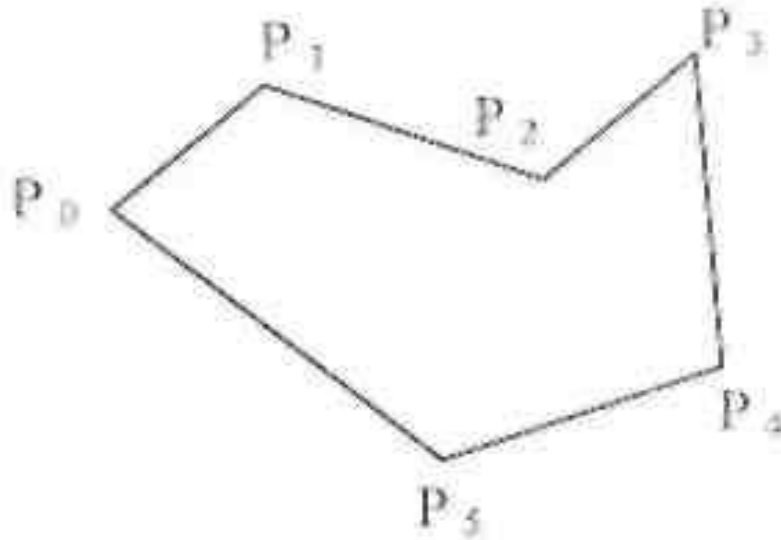
$(x_0, y) - (x_1, y)$

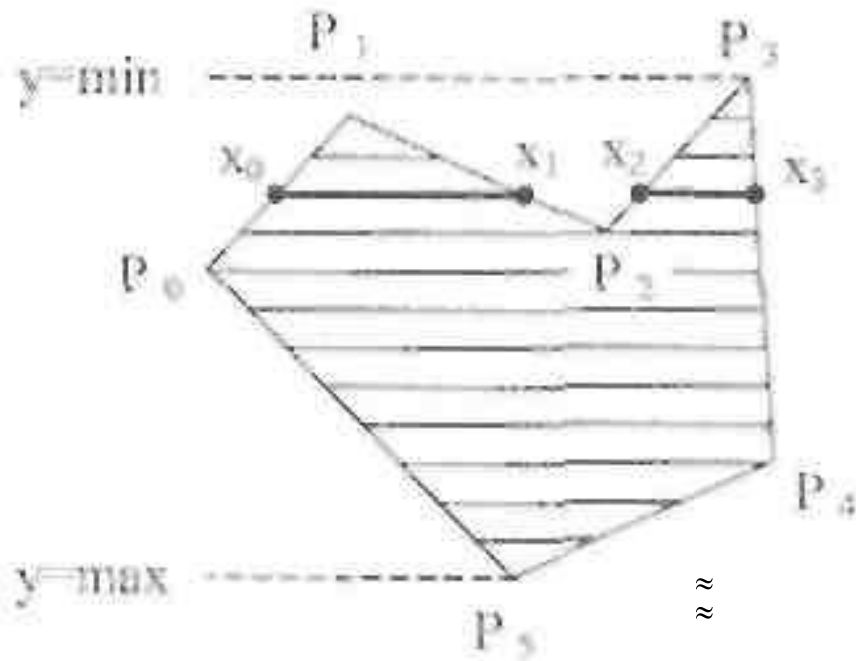
$(x_2, y) - (x_3, y)$

.....

$(x_{2k}, y) - (x_{2k+1}, y)$

Каждый отрезок выводится цветом  
заполнения  
}





$$x = x_i + (y_k - y) (x_k - x_i) / (y_k - y_i)$$

$$N_{\text{ТАКТ}} = (y_{\max} - y_{\min}) N_{\text{ГОР}}$$

$$N_{\text{ГОР}} = kn$$