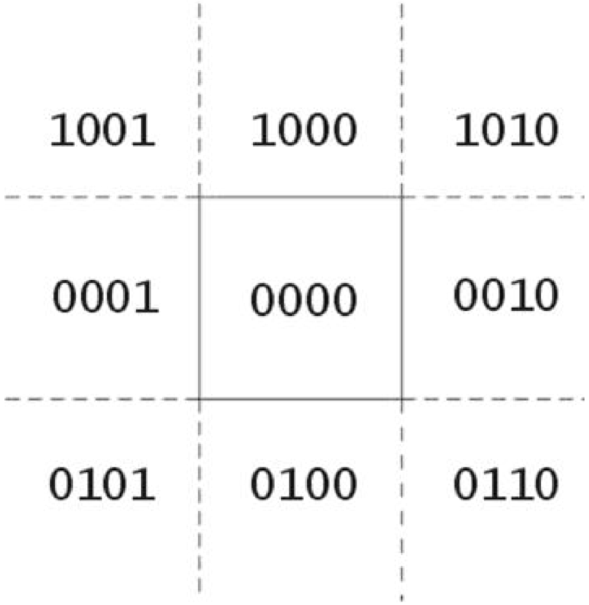
Отсечение по полю вывода
Алгоритм Коэна-Сазерленда

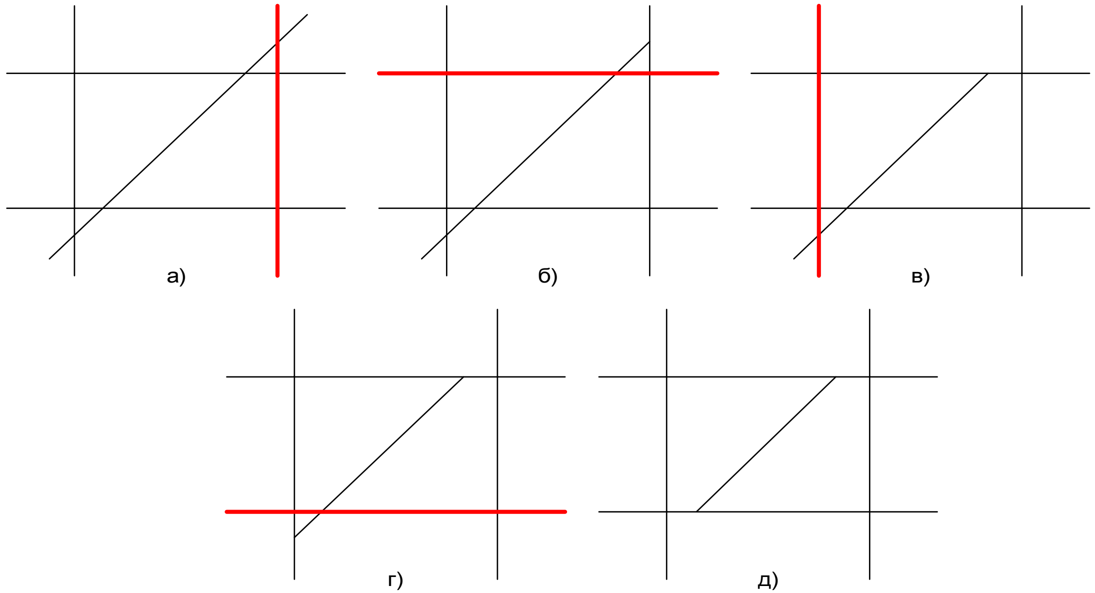


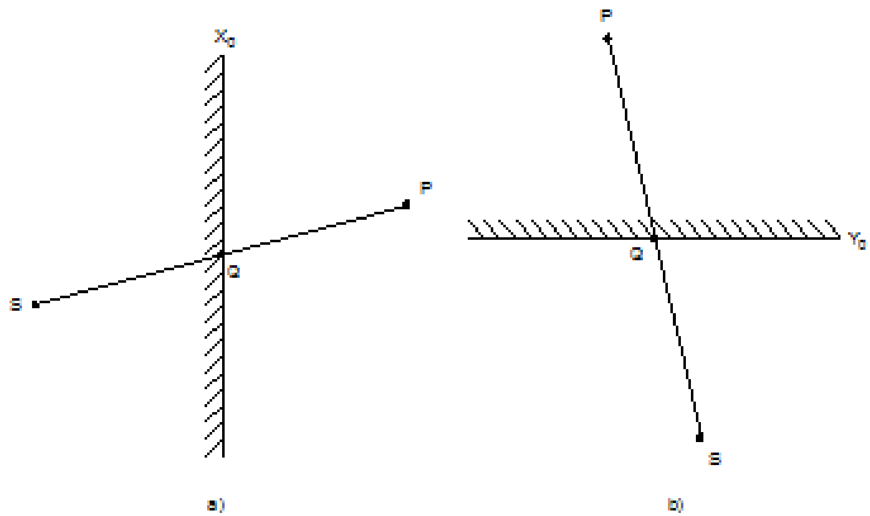
Для каждого отрезка рассчитываются коды концов (К$_1$, К$_2$) затем производится анализ:

– Если $K_1 \wedge K_2 \neq 0$, тогда отрезок лежит вне поля вывода – отрезок отбрасывается
– Если $K_1 = K_2 = 0$, тогда отрезок полностью лежит внутри поля вывода – отсечение не нужно, отрезок полностью прорисовывается
– Если $K_1 \wedge K_2 = 0$, отрезок может частично лежать внутри поля вывода – необходимо отсечение по полю вывода.

Бит 1 – точка находится выше окна;
Бит 2 – точка находится ниже окна;
Бит 3 – точка находится справа от окна;
Бит 4 – точка находится слева от окна;

Алгоритм Коэна-Сазерленда





$$\frac{y_q - y_s}{y_p - y_s} = \frac{x_q - x_s}{x_p - x_s} \quad \rightarrow$$

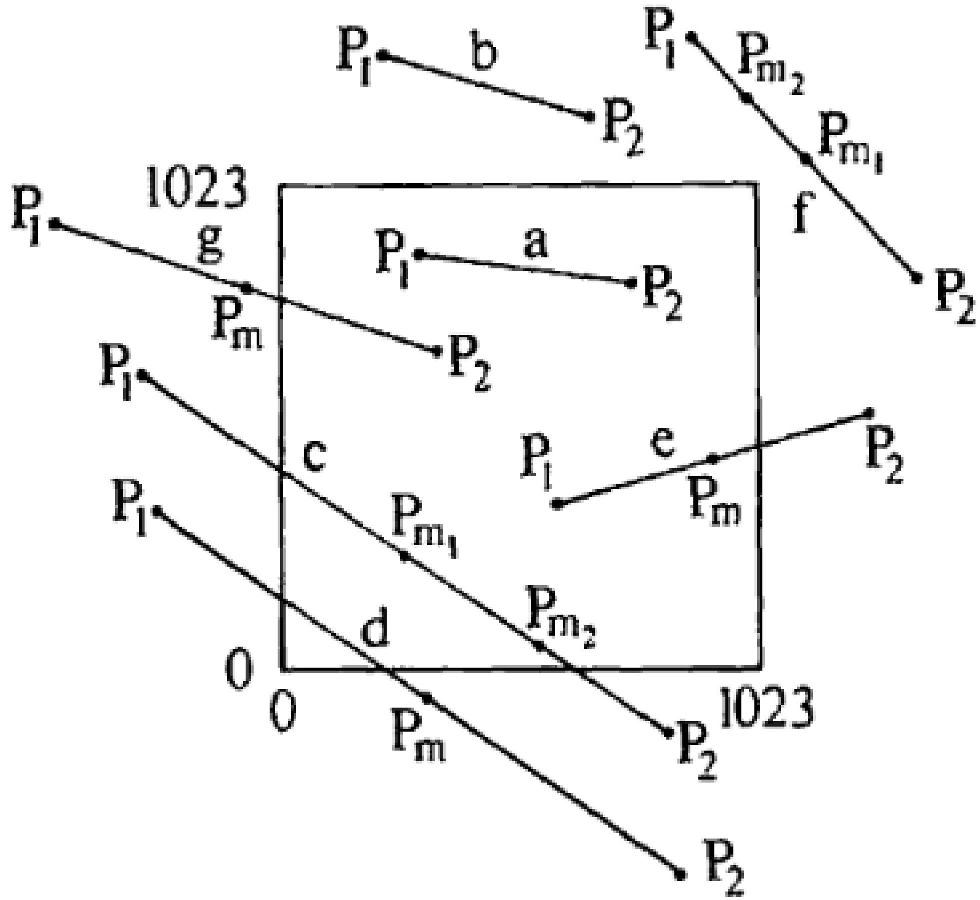
$$y_Q = y_S + \frac{x_0 - x_S}{x_p - x_S} \cdot (y_P - y_S)$$

$$x_Q = x_S + \frac{y_0 - y_S}{y_p - y_S} \cdot (x_P - x_S)$$

Алгоритм разбиения средней точкой

В алгоритме используются коды концевых точек отрезка и проверки, выявляющие полную видимость отрезков, невидимость отрезков.

Те отрезки, которые необходимо отсекать, разбиваются на две равные части. Затем те же проверки применяются к каждой из половин до тех пор, пока не будет обнаружено пересечение со стороной окна или длина разделяемого отрезка не



Алгоритм Кируса – Бека

Параметрическое уравнение отрезка от P_1, до P_2 имеет вид:

$$P(t) = P_1 + (P_2 - P_1)t, \; 0 \leqslant t \leqslant 1.$$

$$x(t) = x_1 + (x_2 - x_1)t \qquad 0 \leq t \leq 1$$

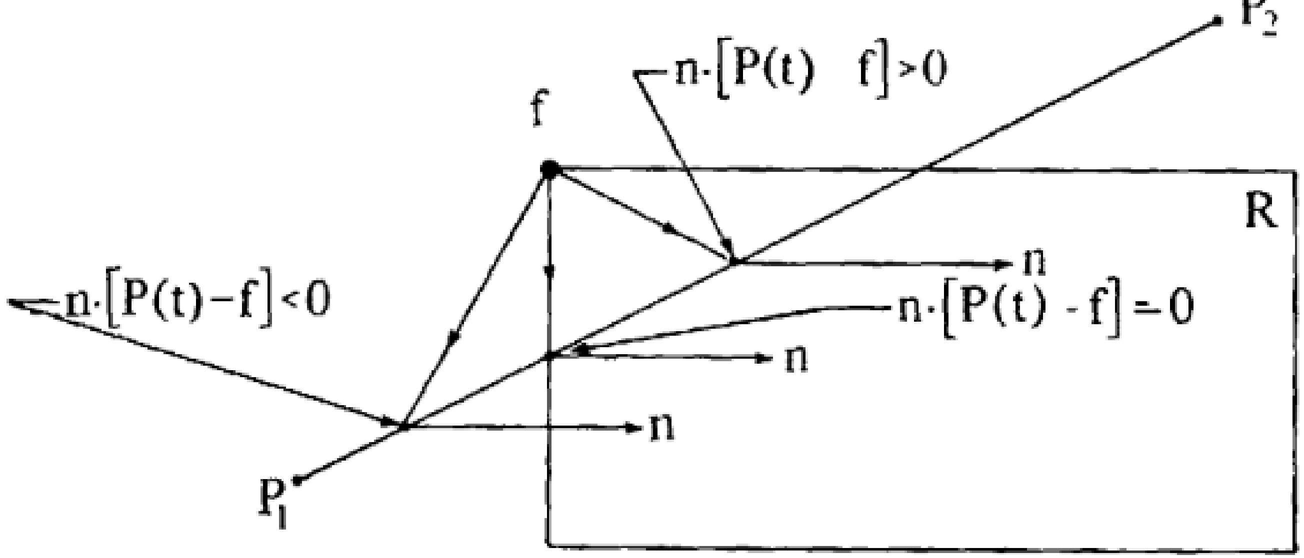
$$y(t) = y_1 + (y_2 - y_1)t \qquad 0 \leq t \leq 1$$

$V_1V_2=|V_1||V_2| \cos\alpha$, где α – это меньший из двух углов.

Если $\alpha=\pi/2$, то $\cos\alpha = 0$ и $V_1 * V_2 = 0$,

$\pi/2 \leq \alpha \leq \pi/2$. При таких значениях угла косинус его всегда положителен.

Алгоритм Кируса – Бека



f – граничная точка выпуклой области *R*,
n – внутренняя нормаль к одной из ограничивающих эту область плоскостей для любой точки отрезка P_1 P_2,

Если $n[P(t) - f] < 0$ то вектор $P(t) - f$ направлен вовне области R.
Если $n[P(t) - f] = 0$ то $P(t) - f$ принадлежит плоскости, которая проходит через f и перпендикулярна нормали.
Если $n[P(t) - f] > 0$ следует, что вектор $P(t) - f$ направлен внутрь R.

Алгоритм Кируса – Бека

$$P(t) = P_1 + (P_2 - P_1)t,\ 0 \leqslant t \leqslant 1.$$

$$\mathbf{n}_i \cdot [\mathbf{P}(\mathbf{t}) - \mathbf{f}_i] \qquad i = 1,\ 2,\ 3, \ldots$$

$$\mathbf{n}_i \cdot [\mathbf{P}_1 + (\mathbf{P}_2 - \mathbf{P}_1)\mathbf{t} - \mathbf{f}_i] = 0 \quad \longrightarrow \quad \mathbf{n}_i \cdot [\mathbf{P}_1 - \mathbf{f}_i] + \mathbf{n}_i \cdot [\mathbf{P}_2 - \mathbf{P}_1]\mathbf{t} = 0$$

$$\mathrm{D} = \mathrm{P}_2 - \mathrm{P}_1, \qquad \mathrm{W}_j = \mathrm{P}_1 - \mathrm{f}.$$

$$\mathbf{t}(\mathbf{n}_i \cdot \mathbf{D}) + \mathbf{w}_i \cdot \mathbf{n}_i = 0$$

$$t = -\frac{\mathbf{w}_i \cdot \mathbf{n}_i}{\mathbf{D} \cdot \mathbf{n}_i} \qquad \mathbf{D} \neq 0 \qquad i = 1, 2, 3, \ldots$$

$$\mathbf{w}_i \cdot \mathbf{n}_i \begin{cases} < 0, & \text{то эта точка вне окна} \\ = 0, & \text{то она на границе окна} \\ > 0, & \text{то она внутри окна} \end{cases}$$

Алгоритм Кируса – Бека

Если значение t лежит за

пределами интервала $0 \le t \le 1$, то можно его проигнорировать.

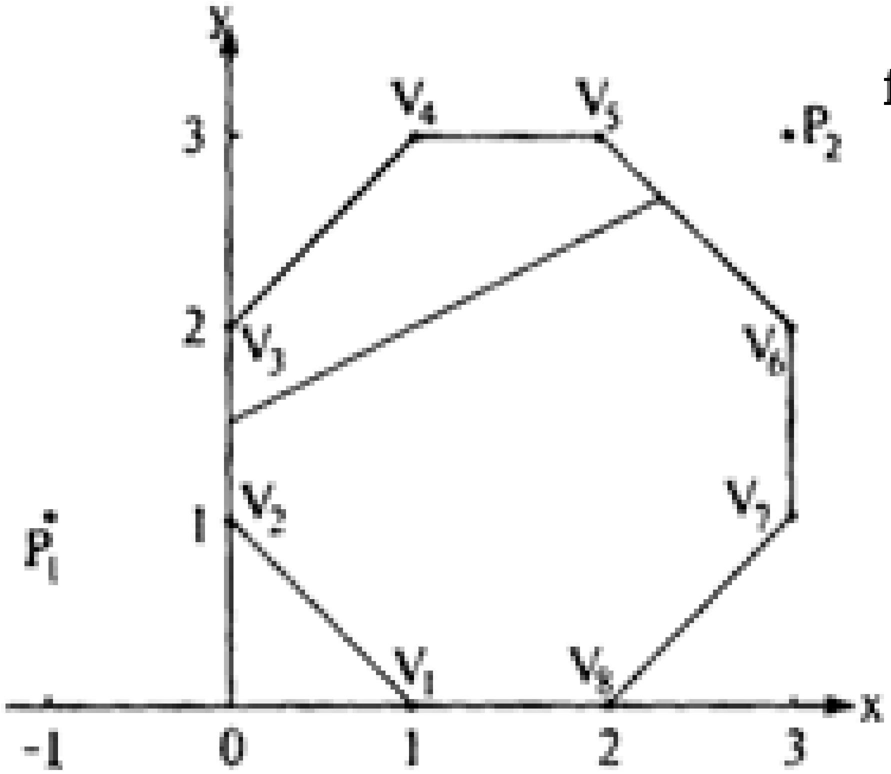
отрезок может пересечь выпуклое окно не более чем в двух точках.

решения следует разбить на две группы: нижнюю и верхнюю, в зависимости от того, к началу или к концу отрезка будет ближе соответствующая точка. Нужно найти наибольшую из нижних и наименьшую из верхних точек. Если $D * n_i > 0$, то найденное значение t рассматривается в качестве возможного нижнего предела. Если $D * n_i < 0$, то значение t рассматривается в качестве возможного верхнего предела.

Алгоритм Кируса – Бека

Отрезок от $P_1(-1, 1)$ до P_2 (3, 3)

$D = P_2 - P_1, = [3\ 3] - [-1\ 1] = [4\ 2]$.



ребро от V_5 до V_6.

f(2, 3) $w = P_1 - f = [-\ 1\ \ 1] - [2\ \ 3] = [-\ 3\ \ -2]$.

Внутренняя нормаль к ребру $I_5\ I_6$ равна $n = [-1 - 1]$. Значит, $D*n = -6 < 0$,

$w*n = 5$,

$t_в = -5\ (-6) = 5/6$.

$$t = -\frac{\mathbf{w}_i \cdot \mathbf{n}_i}{\mathbf{D} \cdot \mathbf{n}_i} \qquad \mathbf{D} \neq 0 \qquad i = 1, 2, 3, \ldots$$

Алгоритм Кируса – Бека

Ребро	n		f	w		w · n	D · n[1]	$t_н$	$t_в$
V_1V_2	[1	1]	(1, 0)	[2	1]	−1	6	1/6	
V_2V_3	[1	0]	(0, 2)	[−1	−1]	−1	4	1/4	
V_3V_4	[1	−1]	(0, 2)	[−1	−1]	0	2	0	
V_4V_5	[0	−1]	(2, 3)	[−3	−2]	2	−2		1
V_5V_6	[−1	−1]	(2, 3)	[−3	−2]	5	−6		5/6
V_6V_7	[−1	0]	(3, 1)	[−4	0]	4	−4		1
V_7V_8	[−1	1]	(3, 1)	[−4	0]	4	−2		2
V_8V_1	[0	1]	(1, 0)	[−2	1]	1	2	−1/2	

максимальное среди значений $t_н$ равно 1/4.

а минимальное среди $t_в$ равно 5/6.

отрезок видим в

интервале $1/4 \le t \le 5/6$. или от точки (0,3/2) до точки (7/3, 8/3).

Определение факта выпуклости многоугольника

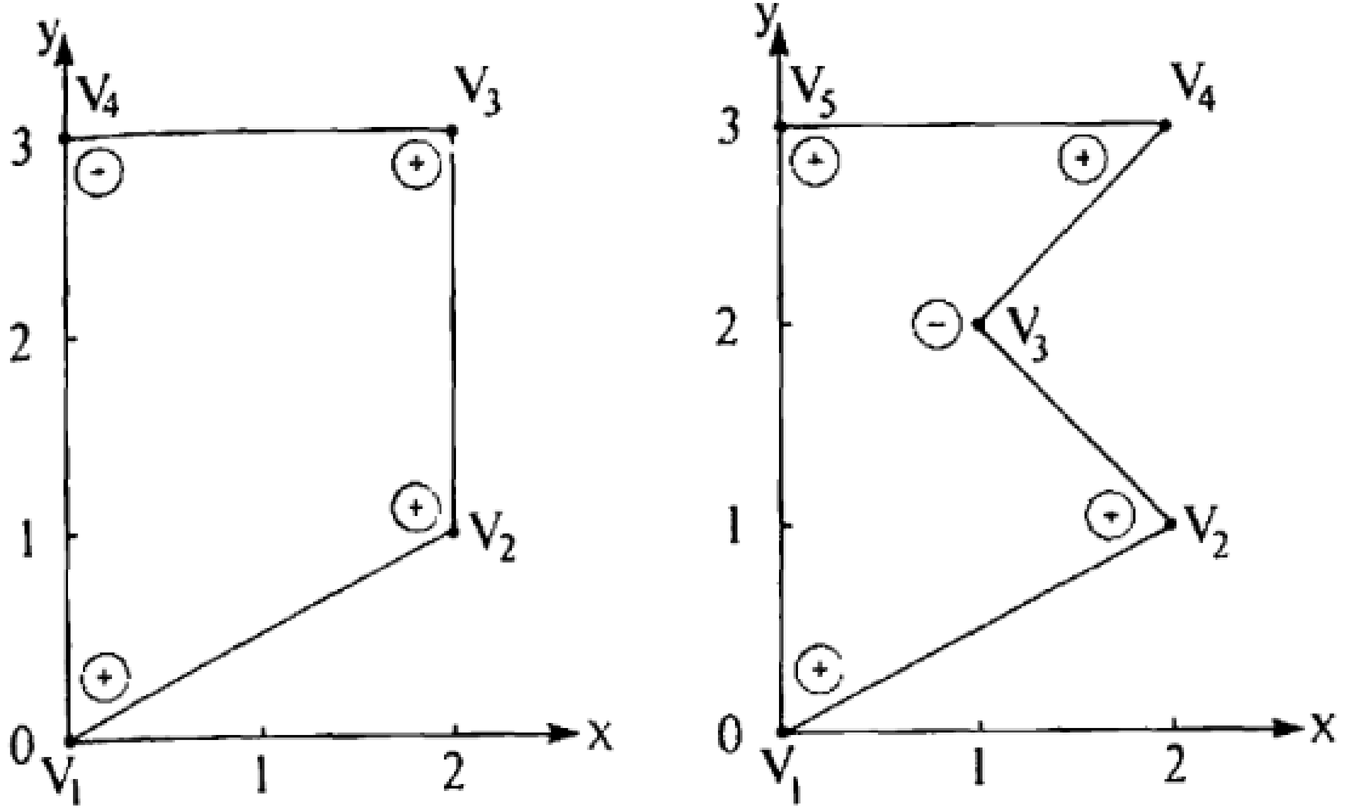
1. Факт выпуклости или не выпуклости двумерного полигонального окна можно установить путем вычисления векторных произведений его смежных сторон. Выводы, которые можно сделать из анализа знаков этих произведений.

2. Другой подход заключается в том, что одна из вершин многоугольника может быть выбрана базой, и могут вычисляться векторные произведения для пар векторов, начинающихся в этой базе и заканчивающихся в последовательных вершинах многоугольника. Результаты этого метода интерпретируются точно так же.

Определение факта выпуклости многоугольника

Результат	Вывод
Все произведения равны нулю	Многоугольник вырождается в отрезок
Есть как положительные, так и отрицательные знаки произведения	Многоугольник невыпуклый
Все знаки произведения неотрицательные	Многоугольник выпуклый, а внутренние нормали ориентированы влево от его контура
Все знаки произведения неположительные	Многоугольник выпуклый, а внутренние нормали ориентированы вправо от его контура.

Определение факта выпуклости многоугольника



Определение нормалей многоугольника

Нормаль к стороне многоугольника можно вычислить, зная что скалярное произведение пары перпендикулярных векторов равно нулю.

Если n_x и n_y – неизвестные компоненты нормали к вектору (V_x, V_y) стороны многоугольника, то

$$\mathbf{n} \cdot \mathbf{V}_e = (n_x\mathbf{i} + n_y\mathbf{j}) \cdot (V_{e_x}\mathbf{i} + V_{e_y}\mathbf{j}) = n_x V_{e_x} + n_y V_{e_y} = 0$$

$$n_x V_{e_x} = -n_y V_{e_y}$$

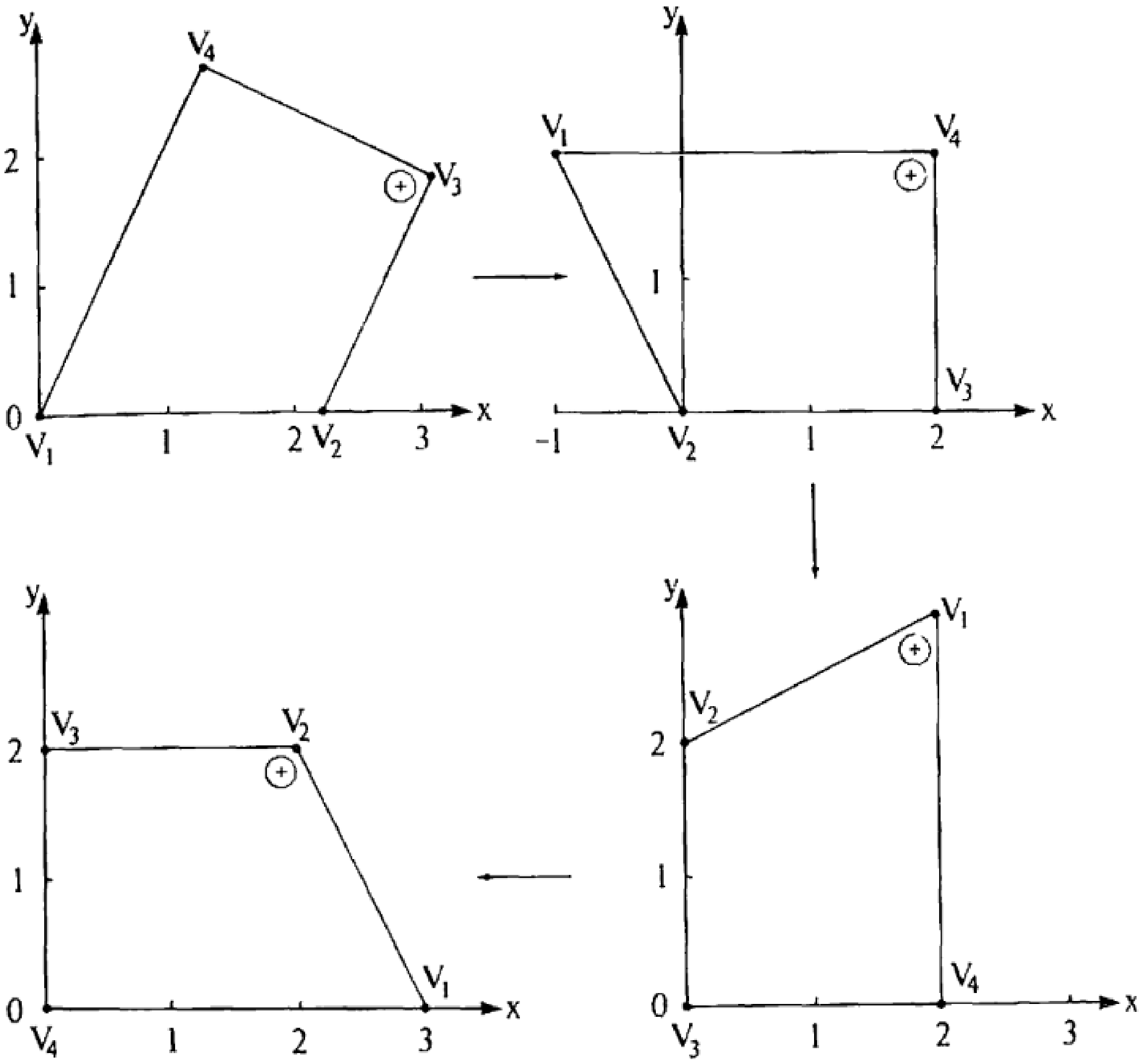
Поскольку нас интересует только направление нормали, то пусть n_y = 1 без потери общности.

$$\mathbf{n} = -V_{e_y}/V_{e_x}\mathbf{i} + \mathbf{j}.$$

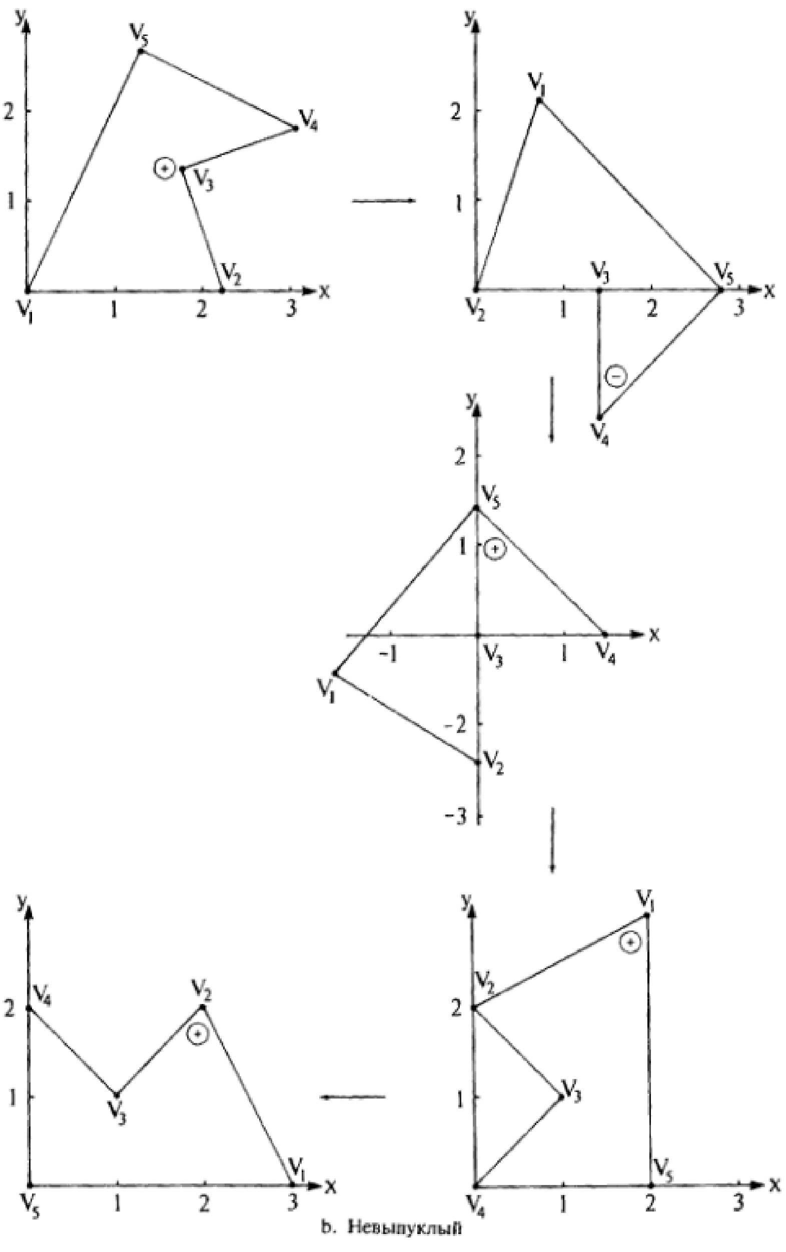
3. Определение факта выпуклости многоугольника

1. Для каждой i-й вершины полигонального окна надо перенести его так, чтобы эта вершина совпала с началом координат.
2. Повернуть многоугольник относительно начала координат так, чтобы (i + 1)-я его вершина оказалась на положительной полуоси X.
3. Вычислить знак ординаты у (i + 2)-й вершины.
4. Если знаки ординат у всех (i + 2)-х вершин совпадают, то окно выпукло; иначе, оно невыпукло.
6. В повернутой системе координат внутренняя нормаль к (i + 1)-й стороне выпуклого многоугольника имеет нулевую абсциссу и ординату, равную знаку ординаты (i + 2)-й вершины.

Определение выпуклости



а. Выпуклый



Определение факта выпуклости многоугольника

Разбиение невыпуклых многоугольников

Если вершины многоугольника перечисляются против часовой стрелки, то процедура будет иметь такой вид:

1. Для каждой i-й вершины многоугольника надо так его перенести, чтобы она совпадала с началом координат.
2. Повернуть многоугольник относительно координат по часовой стрелке так, чтобы (i + 1)-я его вершина оказалась на положительной полуоси x.
3. Проанализировать знак ординаты (i + 2)-й вершины. Если он неотрицателен, то многоугольник выпуклый в (i + 1)-й вершине. Если же этот знак отрицателен, то многоугольник невыпуклый; необходимо разбить его.

Разбиение невыпуклых многоугольников

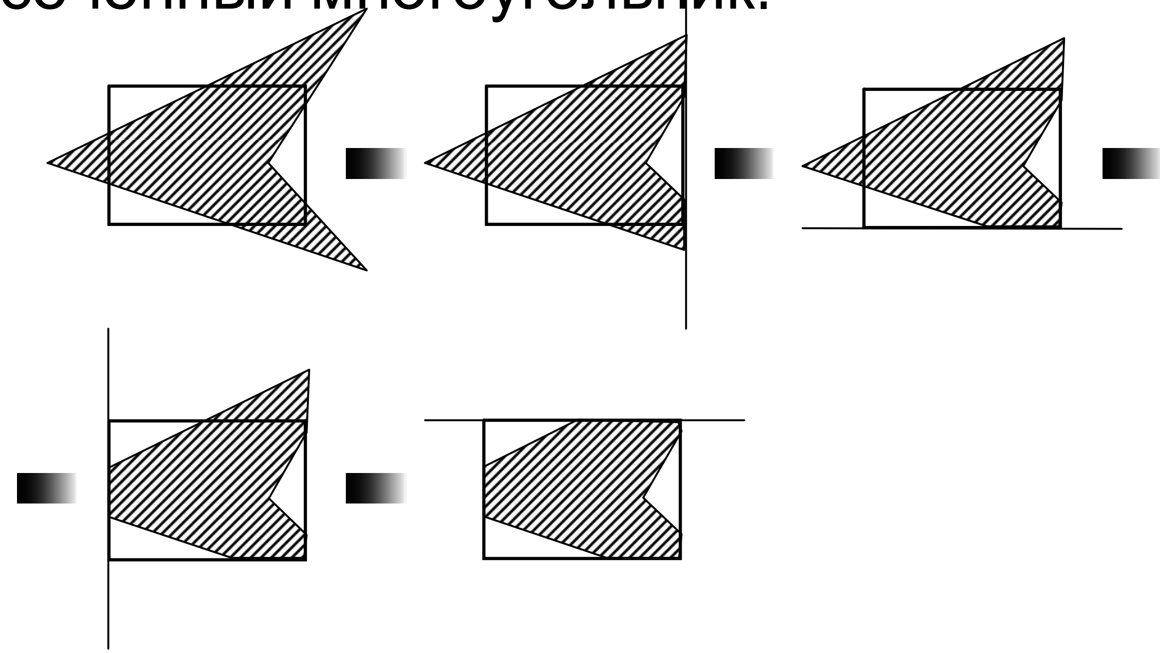
4. Многоугольник разрезается вдоль положительной полуоси x, т. е. ищутся все такие его стороны, которые пересекаются с осью x. Образуются два новых многоугольника: один состоит из вершин, лежащих выше оси x и ближайшей к началу координат точки пересечения с $x > x_{i + 1}$,а второй – из вершин, лежащих ниже оси x и уже упомянутой точки пересечения.

5. Алгоритм рекурсивно применяется к полученным многоугольникам до тех пор, пока все они не станут выпуклыми.

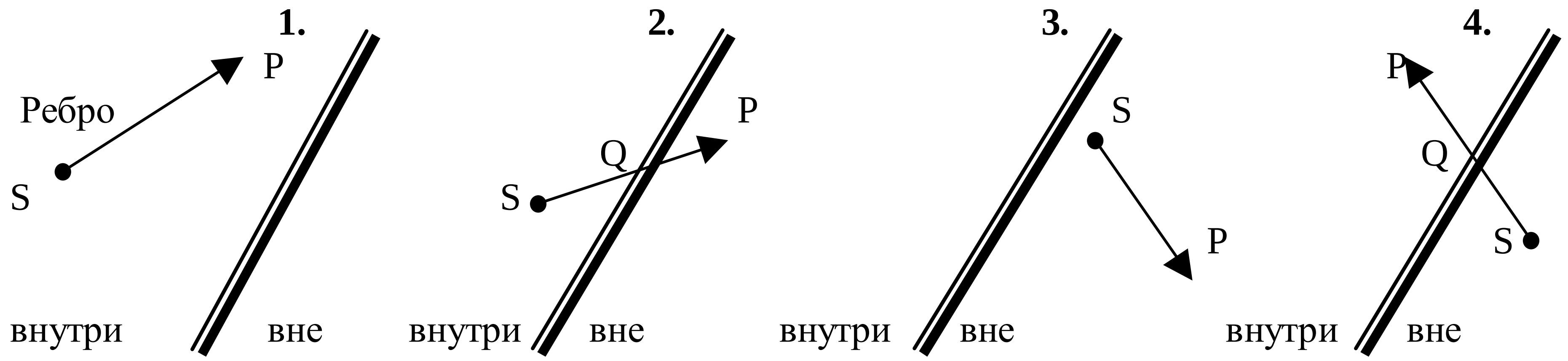
Этот алгоритм не дает оптимального разбиения в смысле минимального числа выпуклых компонент.

Алгоритм Сазерленда-Ходгмана

На вход алгоритма поступает последовательность вершин многоугольника V_1, V_2, ..., V_n. Ребра многоугольника проходят от V_i к V_{i+1} от V_n к V_1. С помощью алгоритма производится отсечение относительно ребра и выводится другая последовательность вершин, описывающая усеченный многоугольник.



Алгоритм Сазерленда-Ходгмана



В случае 1: Добавить в список рисуемых вершин P;

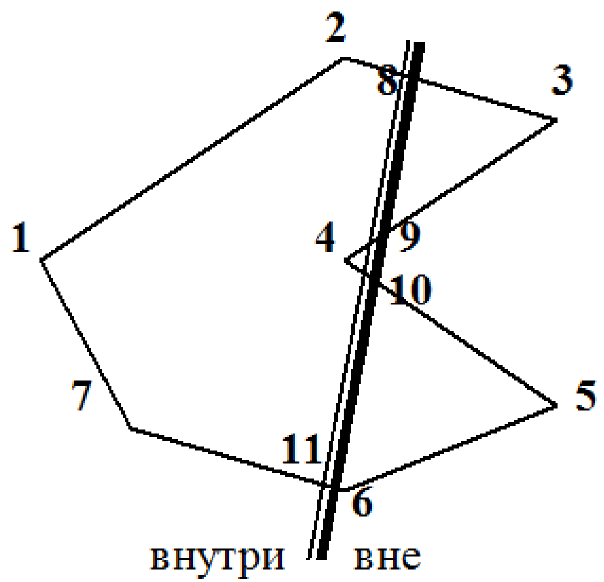
2: Добавить в список Q;

3: Ничего не добавляется;

4: Добавить в список P,Q;

Алгоритм Сазерленда-Ходгмана

Пример: многоугольник задан списком вершин {1,2,3,4,5,6,7,1}.
Последовательно переберём все рёбра.
Начнем процесс с точки 1 и ребра 1-2. Начнём формировать новый список вершин.



В соответствии с ориентацией ребра занесём в выходной список вершину 2 => {2}.
Далее рассмотрим ребро 2-3: добавляется точка 8 => {2, 8}.
3-4: {2, 8, 9, 4}.
4-5: {2, 8, 9, 4, 10}.
5-6: {2, 8, 9, 4, 10}. Ребро полностью оказалось вне, поэтому ничего не добавляется.
6-7: {2, 8, 9, 4, 10, 11, 7}.
7-1: {2, 8, 9, 4, 10, 11, 7, 1}.
Таким образом,
новый список вершин: {2, 8, 9, 4, 10, 11, 7, 1}.

Алгоритм Вейлера – Азертона

Как обрабатываемый (объект), так и отсекающий (отсекатель) многоугольники описываются в алгоритме циклическими списками их вершин. Внешняя граница каждого из этих многоугольников обходится по *часовой стрелке*, а внутренние границы или отверстия – *против часовой* стрелки.

Границы обрабатываемого (объекта) и отсекающего (отсекателя) многоугольников могут пересекаться или не пересекаться между собой. Если они пересекаются, то точки пересечения образуют пары. Одно пересечение из пары возникает, когда ребро обрабатываемого многоугольника входит внутрь отсекающего многоугольника, а другое – когда оно выходит оттуда.

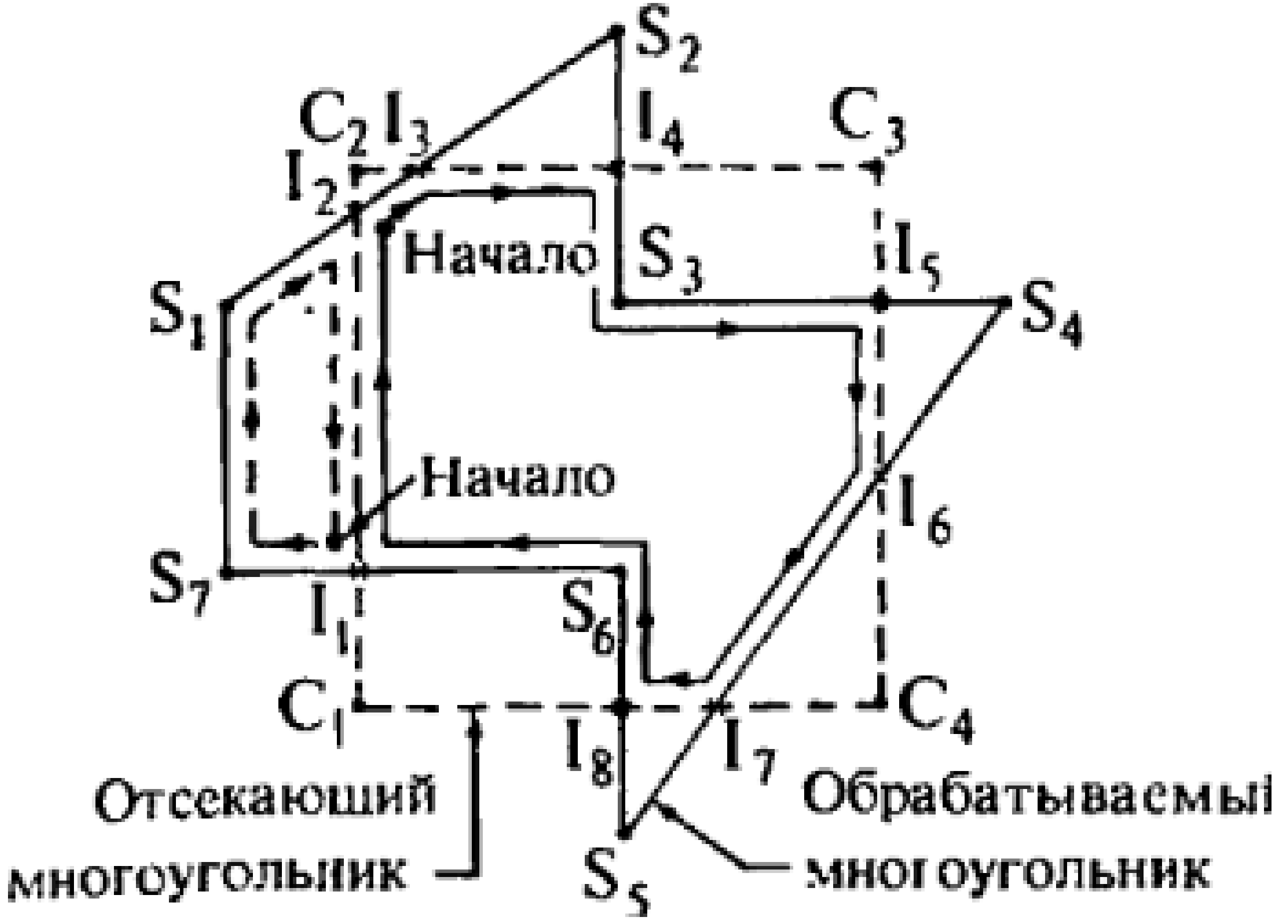
Алгоритм Вейлера – Азертона

Алгоритм начинает с точки пересечения входного типа, затем он прослеживает внешнюю границу по часовой стрелке до тех пор, пока не обнаруживается еще одно ее пересечение с отсекающим многоугольником. В точке последнего пересечения производится поворот направо и далее прослеживается внешняя граница отсекателя по часовой стрелке до тех пор, пока не обнаруживается ее пересечение с обрабатываемым многоугольником.

Этот процесс продолжается до тех пор, пока не встретится начальная вершина.

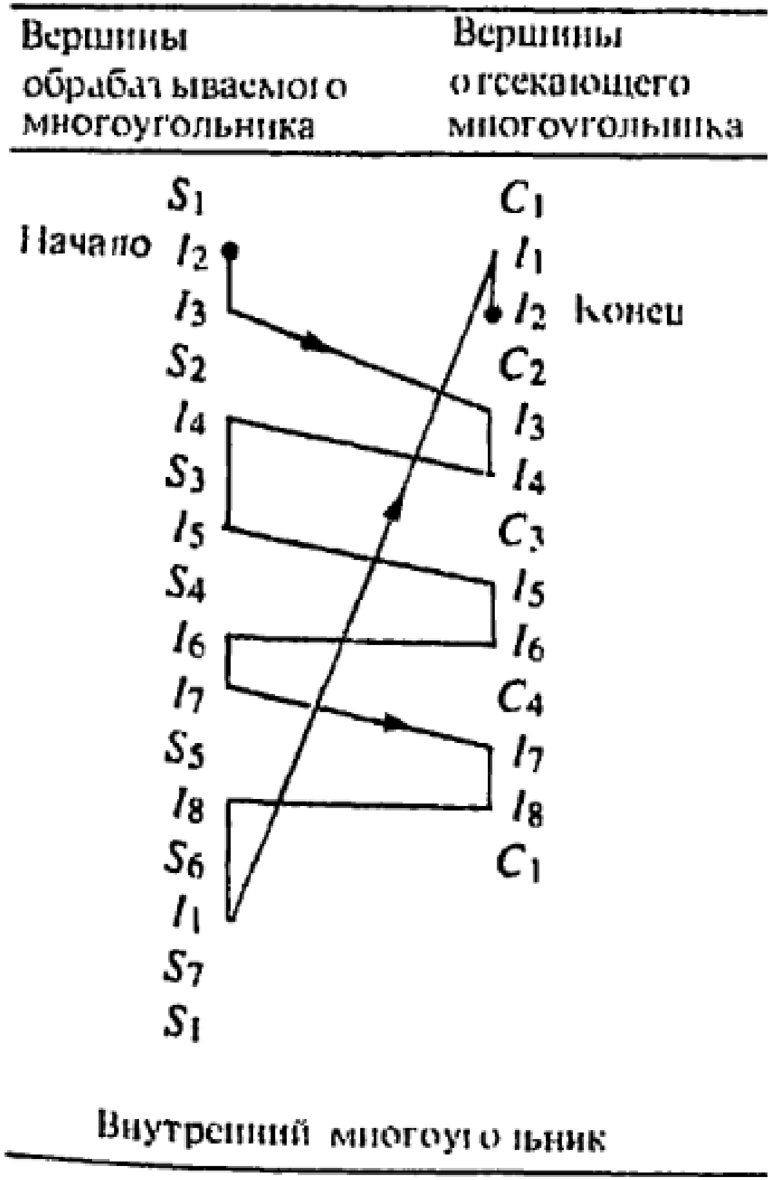
Внутренние границы обрабатываемого многоугольника обходятся против часовой стрелки

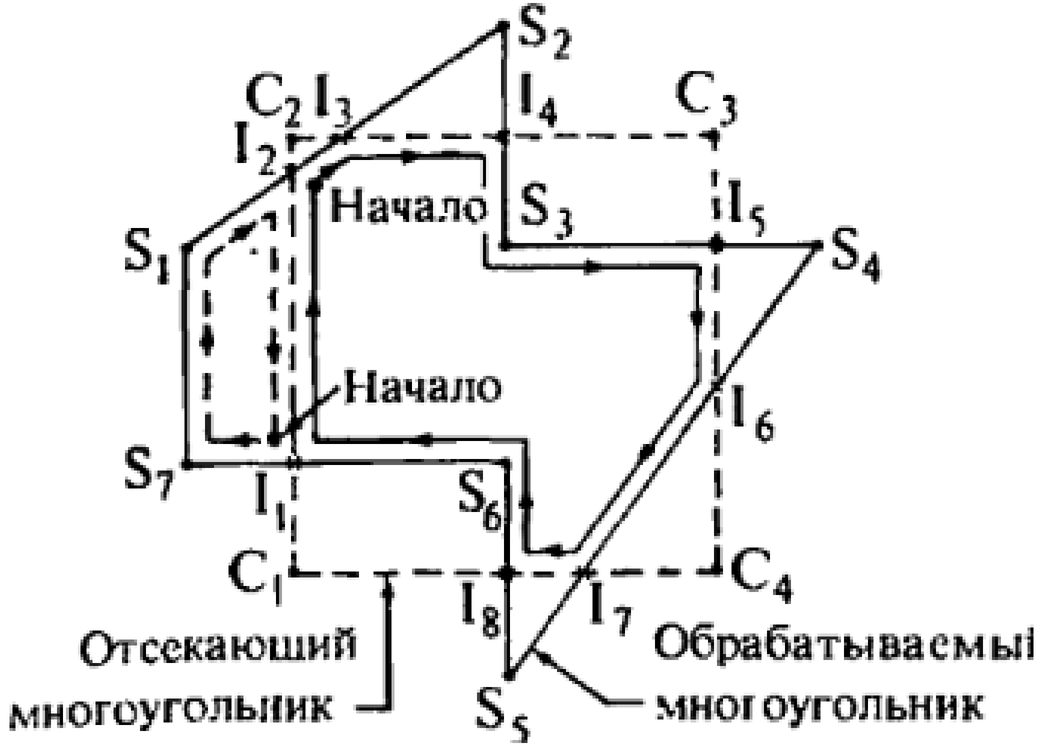
Алгоритм Вейлера – Азертона



Ниже приводятся списки вершин обрабатываемого и отсекающего многоугольников.
В список входов занесены вершины I_2, I_4, I_6 и I_8,
а в список выходов — вершины I_1, I_3, I_5, I_7.

Алгоритм Вейлера – Азертона

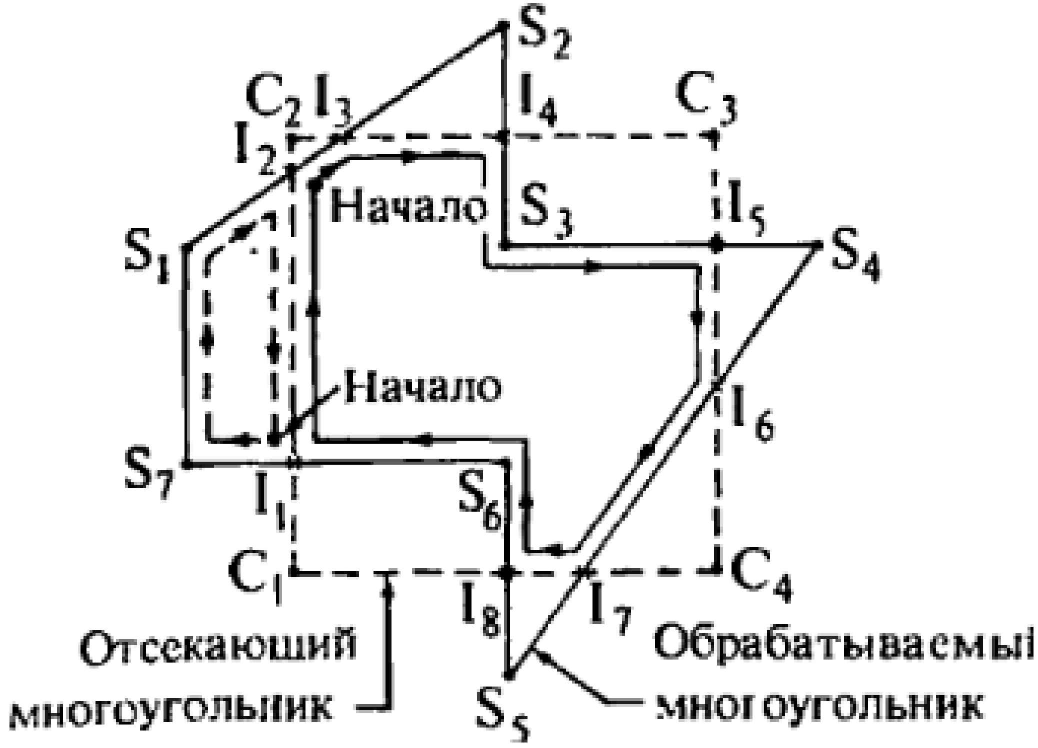




$I_2 I_3 I_4 S_3 I_5 I_6 I_7 I_8 S_6 I_1 I_2.$

Алгоритм Вейлера – Азертона

Вершины обрабатываемого многоугольника	Вершины отсекающего многоугольника
S_1	C_1
I_2	I_1 Конец
I_3	I_2
S_2	C_2
I_4	I_3
S_3	I_4
I_5	C_3
S_4	I_5
I_6	I_6
I_7	C_4
S_5	I_7
I_8	I_8
S_6	C_1
I_1 Начало	
S_7	
S_1	

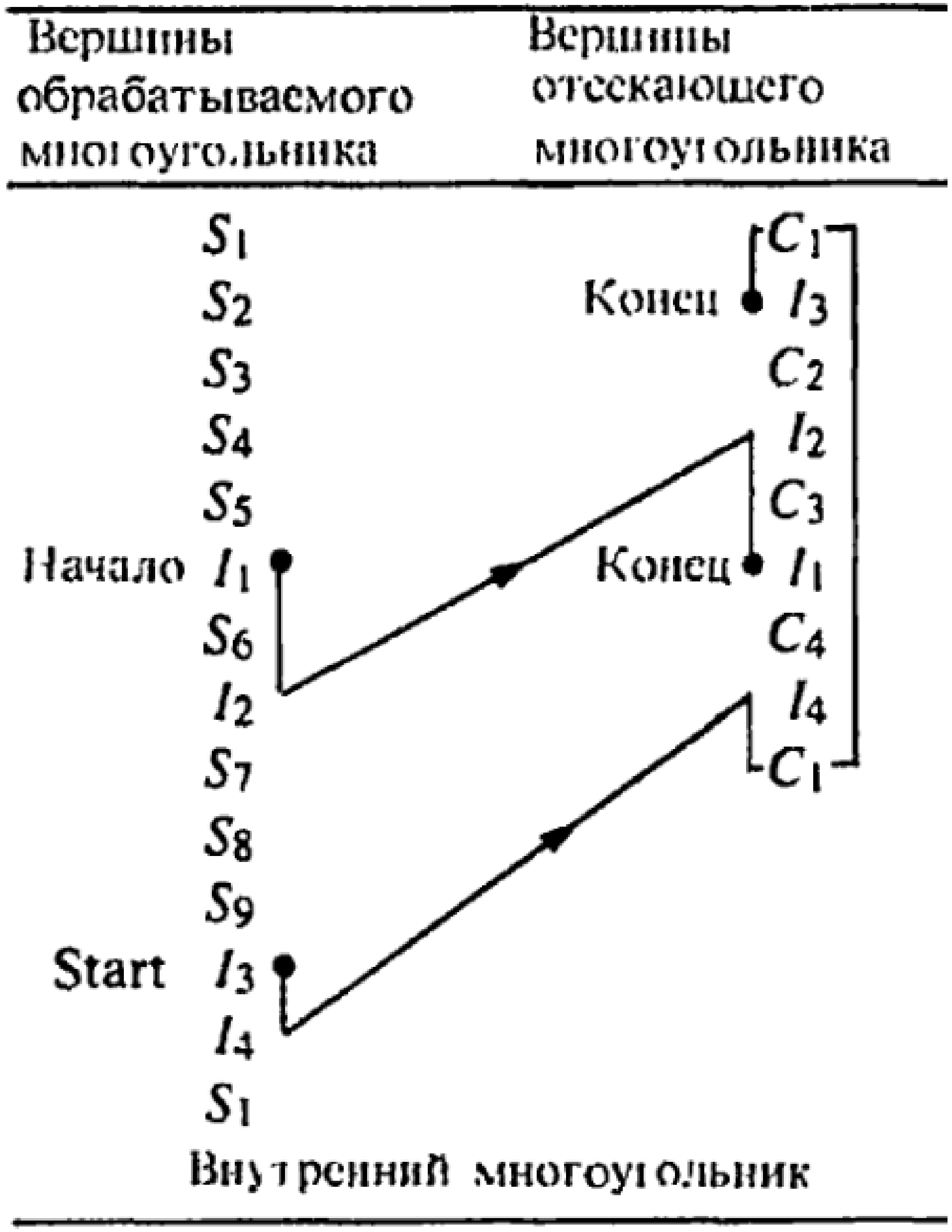


$$I_1 S_7 S_1 I_2 I_1.$$

Алгоритм Вейлера – Азертона

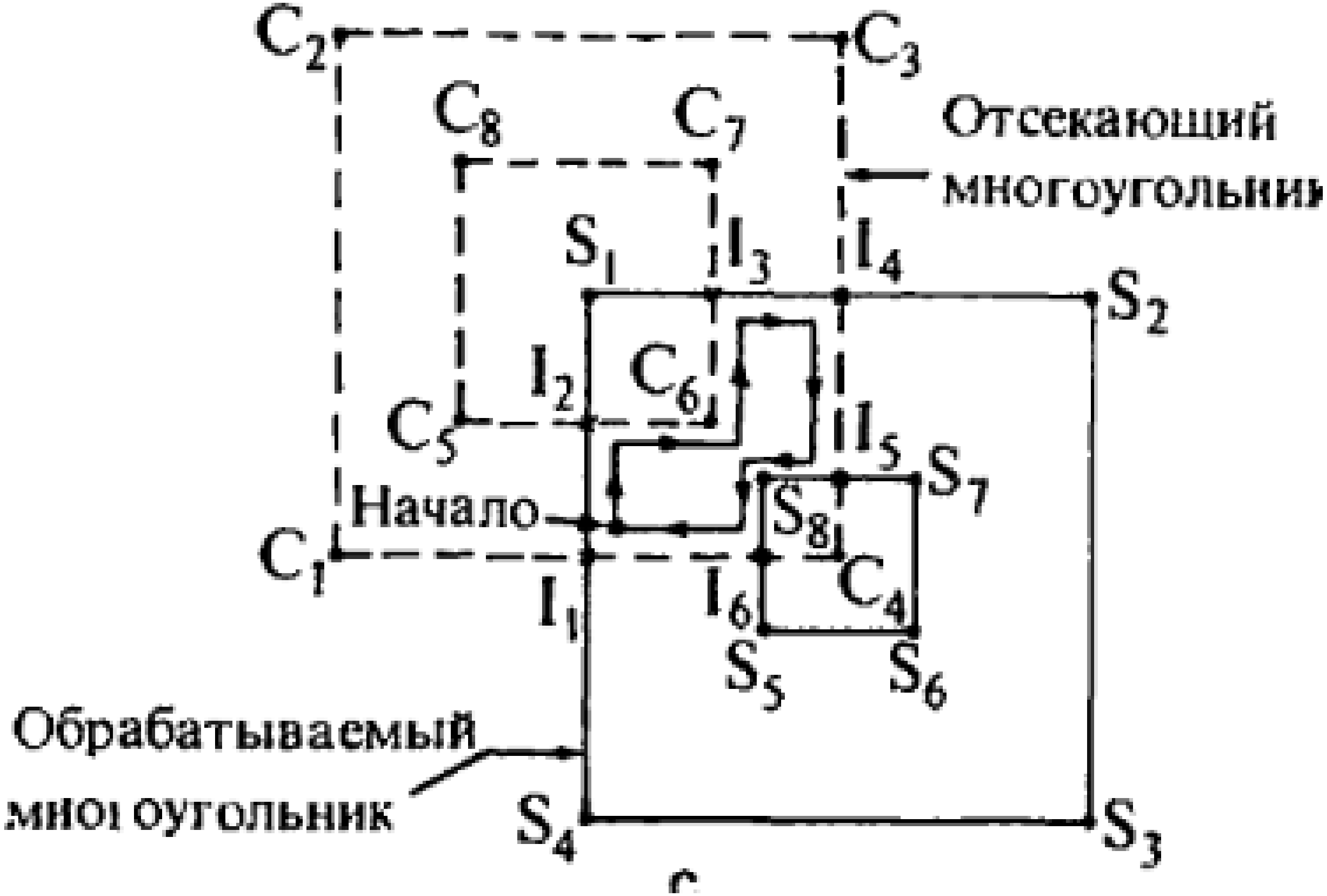
Точки пересечения I_1, и I_3 попадают в список входов,
а I_2 и I_4 — в список выходов.

Алгоритм Вейлера – Азертона



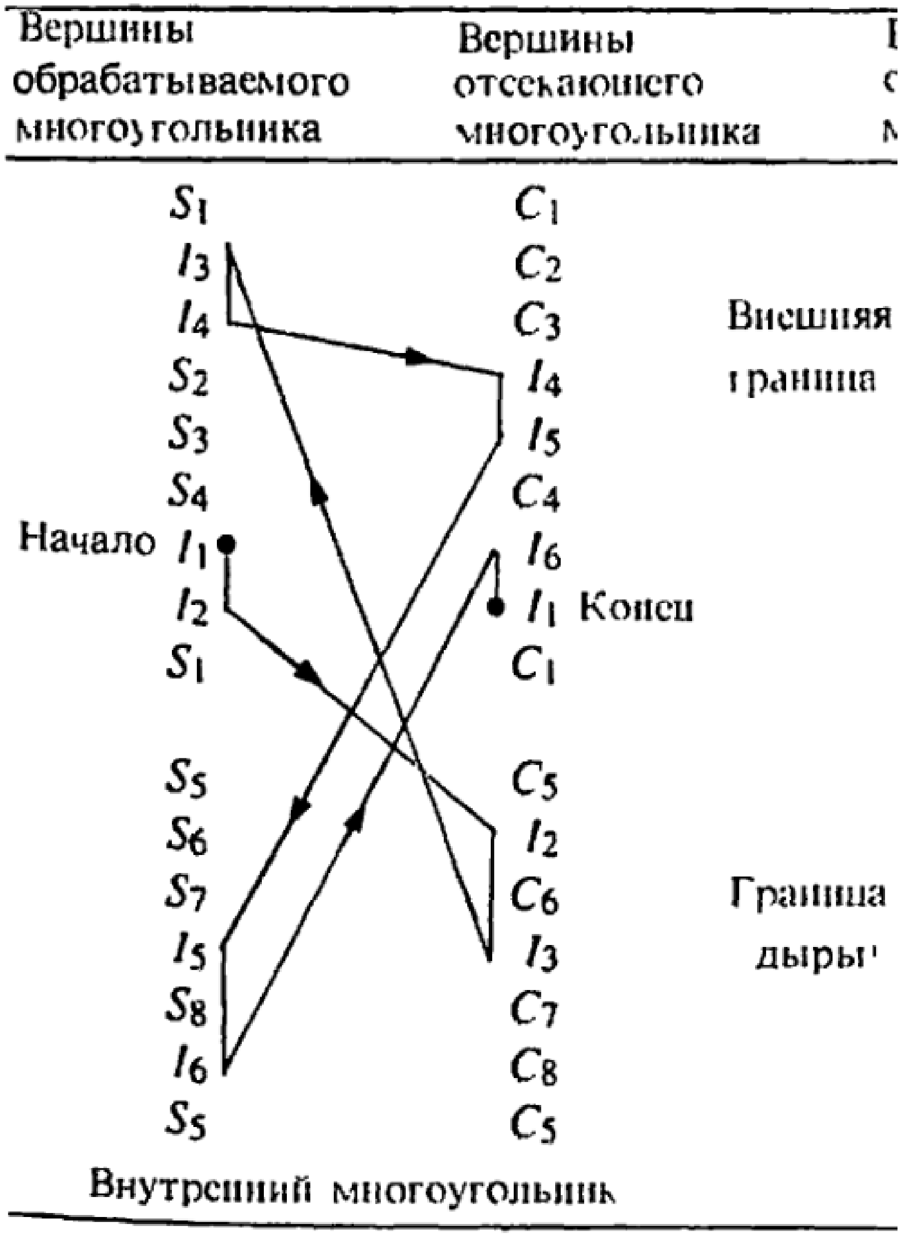
$I_1 S_6 I_2 C_3 I_1$ и $I_3 I_4 C_1 I_3$.

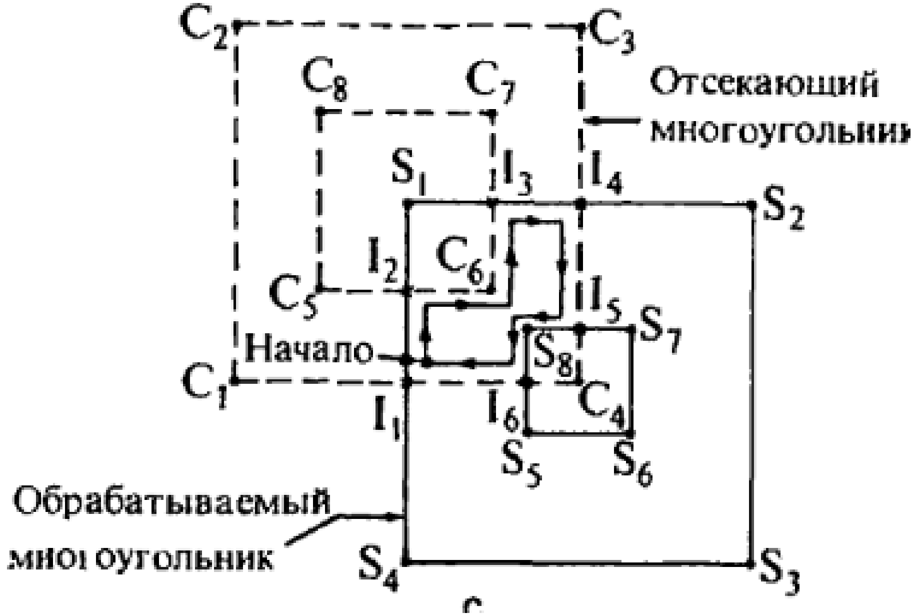
Алгоритм Вейлера – Азертона



Точки пересечения I_1 I_3 I_5 помешаются в список входов, а I_2, I_4. и I_6. — в список выходов.

Алгоритм Вейлера – Азертона

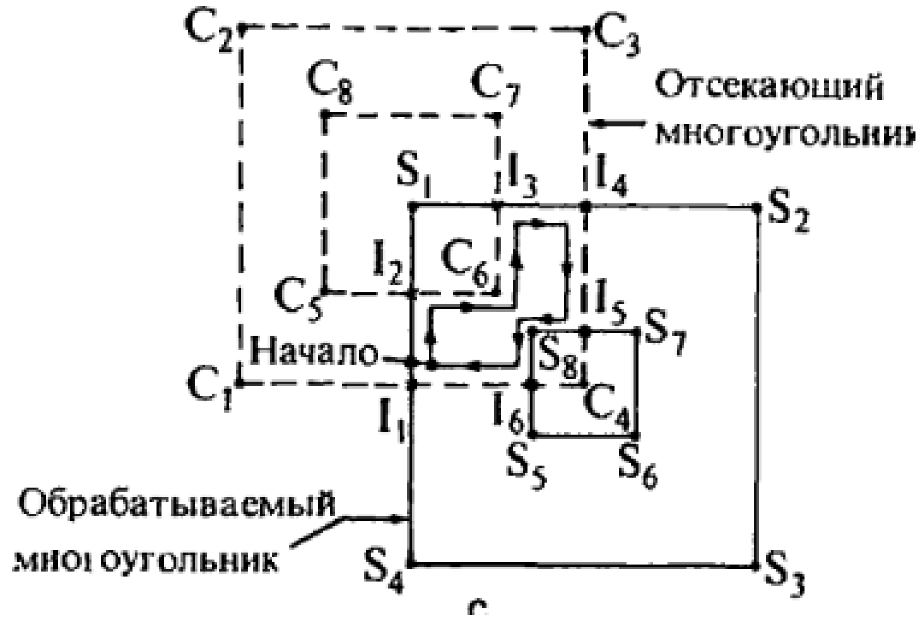




$$I_1 I_2 C_6 I_3 I_4 I_5 S_8 I_6 I_1$$

Алгоритм Вейлера – Азертона

Вершины обрабатываемого многоугольника		Вершины отсекающего многоугольника	
S_1		C_1	
I_3		C_2	
I_4		C_3	
S_2		I_4	
S_3		I_5	
S_4		C_4	
I_1		I_6	
I_2	Начало	I_1	
S_1		C_1	
S_5		C_5	
S_6		I_2	Конец
S_7		C_6	
I_5		I_3	
S_8		C_7	
I_6		C_8	
S_5		C_5	
Внешний многоугольник			



$$I_2 S_1 I_3 C_6 I_2$$