5. OREZÁVANIE ÚSEČKY

Orezávanie bodu je pomerne jednoduché. Na zložitejšie objekty však potrebujeme viac operácií, a preto sa snažíme ich počet minimalizovať. Tak je to aj s úsečkou vo všeobecnej polohe. Najprv otestujeme jej krajné body, a ak patria do okna, tak aj celá úsečka leží v okne. V prípade, že úsečka neleží v okne, otestujeme jej prienik s oknom. Ak úsečka pretína okno, tak ju orezávame, a na to potrebujeme zistiť body prieniku (t.j. kde úsečka pretne hranice okna).

Podstatnou vlastnosťou pri orezávaní je rýchlosť algoritmu na nájdenie bodov prieniku (pri vykresľovaní scény môžeme mať rádovo tisíce objektov, ktoré treba orezať za čo najkratší čas). Preto vzniklo niekoľko algoritmov na orezávanie. Najprv ale musíme urobiť testy, či naozaj treba nájsť body prieniku. Úsečky, ktoré ležia v okne orezávať netreba, tie môžeme rovno vykresliť. Ak oba krajné body úsečky ležia nad oknom, t.j. majú súradnicu $y > y_{max}$ okna, tak úsečku nevykreslíme. Podobne ak oba krajné body úsečky ležia pod oknom, naľavo či napravo od okna, tak úsečku nevykreslíme.

Algoritmus orezávania Cohen-Sutherlanda

Prvá časť tohto algoritmu slúži na vylúčenie úsečiek, u ktorých netreba počítať prienik s oknom. Najprv rozdelíme rovinu na deväť častí, pričom prostredná časť bude predstavovať okno. Každej časti priradíme 4-bitový kód, ktorý dostane aj každý bod úsečky, ktorý do tejto oblasti padne.

Jednotlivé bity pre bod (x, y) sa nastavia nasledovne:

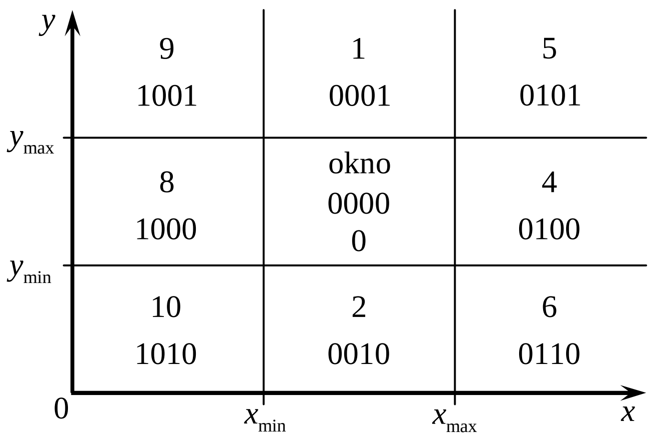
1= 1. bit- bod leží naľavo od okna ($x_{min} > x$)

1= 2. bit- bod leží napravo od okna ($x_{max} < x$)

1= 3. bit- bod leží nižšie od okna ($y_{min} > y$)

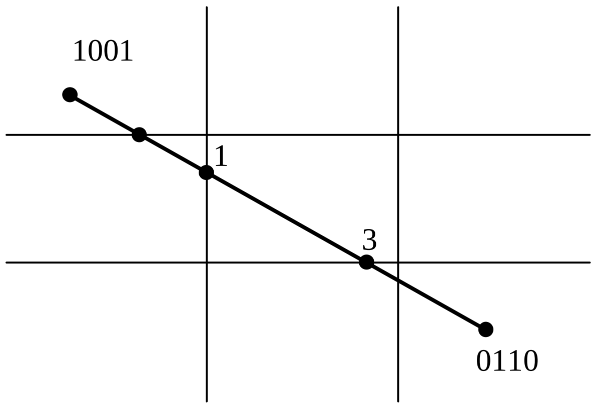
1= 4. bit- bod leží vyššie od okna ($y_{max} < y$)

Napríklad: 0101 – bod leží napravo a vyššie od okna



Obr. Kódy oblastí určené oknom

Bod, ktorý sa nachádza vo vnútri okna, má kód 0000. Úsečka je celá v okne a môžeme ju vykresliť, ak oba jej krajné body majú kód 0000. Môžeme ju vylúčiť bez vykreslenia, ak oba jej krajné body sú vľavo, vpravo, hore alebo dole od okna. Ľahko to identifikujeme práve podľa kódov krajných bodov úsečky tak, že urobíme ich logický súčin ((1 and 1)=1, inak 0). Ak je rôzny od 0000, tak úsečka nepretína okno a teda ju môžeme vylúčiť z vykresľovania. V opačnom prípade úsečku nemôžeme vylúčiť z vykresľovania a musíme ju otestovať na prienik vzhľadom na hranicu okna, v najhoršom prípade 4-krát.



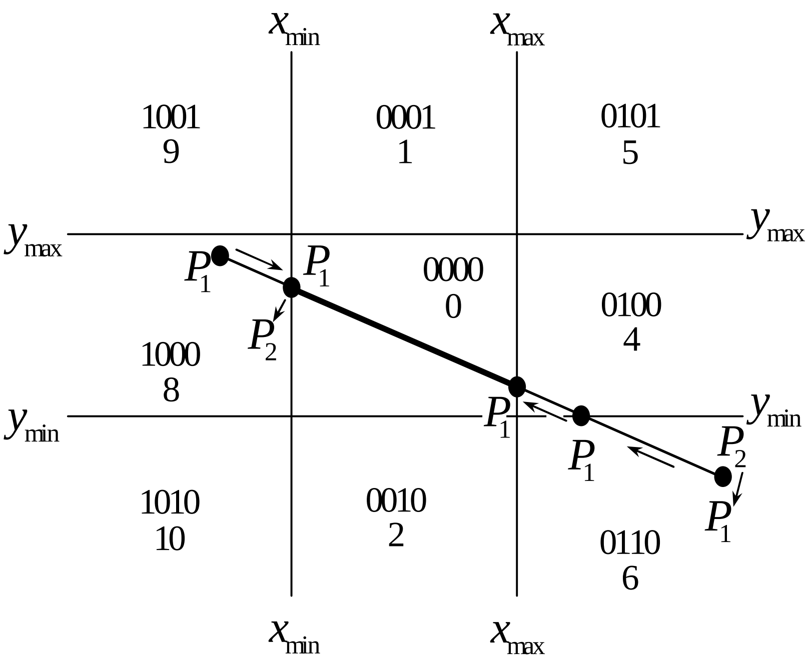
Obr. Orezávanie zadanej úsečky vzhľadom na okno a určenie bodov prieniku

Poznámky k algoritmu:
Súradnice (x_{min}, x_{max}, y_{min}, y_{max}) definujú okno, vzhľadom na ktoré budeme orezávať úsečky.
Krajné body úsečky sú $P_1=(x_1,y_1)$ a $P_2=(x_2,y_2)$.
Procedúra *outcod* nastaví bity do kódu podľa súradníc daného bodu vzhľadom na okno.
Boolovská premenná *accept* nás informuje, či daná úsečka sa má vykresľovať.
Premenná *done* slúži na ukončenie cyklu **repeat...until** .
Funkcia *and* dáva logický súčin kódov.
V 3.kroku algoritmus zisťuje, či úsečka nie je mimo okna podľa spomenutých kritérií a v 4. kroku, či úsečka je vnútri okna.
Procedúra *swap* vymieňa body a ich kódy v 5. kroku.
Postupne od 6. až po 9. krok zisťujeme, či prvý krajný bod P_1 úsečky je zhora, zdola,sprava resp. zľava od okna. Ak áno, tak úsečku P_1 P_2 orežeme hornou, dolnou, pravou resp. ľavou hraničnou priamkou a odstránime z nej tú časť, ktorá obsahuje bod P_1. Vzájomnou výmenou bodov P_1, P_2 orežeme do okna aj „prečnievajúcu“ časť úsečky pri bode P_2.

Algoritmus Cohen-Sutherlanda

```
Procedure Clipping;
begin
1.  accept= false;                                    { nastav - P1P2 sa nevykresľuje }
    outcod (x2, y2, cd2);                             { zistíme kód bodu P2 }
2.  repeat
        outcod (x1, y1, cd1);                         { zistíme kód bodu P1 }
3.      if and (cd1; cd2) ≠ 0 then done:= true        { 1. jednoduchý test - mimo okna}
        else
4.          begin
            if (cd1=0 and cd2=0) then
                        begin  accept:= true;         { žiadaj vykreslenie v kroku 10. }
                               done:= true end
            else
5.              begin
                if cd1=0 then swap(P1, P2);           { zameníme body, aby 1. bol von}
6.              if cd1 ∈ (1, 5, 9) then               { orezávanie zhora }
                    begin
                    x1:= x1 + (x2-x1)*(ymax-y1)/(y2-y1);
                    y1:= ymax ;
                    end
7.              else if cd1 ∈ (2, 6, 10) then         { orež zdola }
                    begin
                    x1:= x1 + (x2-x1)*(ymin-y1)/(y2-y1);
                    y1:= ymin ;
                    end
8.              else if cd1 ∈ (4, 5, 6) then          { orež sprava }
                    begin
                    y1:= y1 + (y2-y1)*(xmax-x1)/(x2-x1);
                    x1:= xmax ;
                    end
9.              else if cd1 ∈ (8, 9, 10) then         { orež zľava }
                    begin
                    y1:= y1 + (y2-y1)*(xmin-x1)/(x2-x1);
                    x1:= xmin ;
                    end
                end              { od kroku 5 }
            end              { od kroku 4 }
    until done
10. if accept then draw(P1, P2);                      { vykresli úsečku }
end.
```

Podľa tohto algoritmu je orezaná úsečka P_1P_2 do okna $(x_{min}, y_{min}) – (x_{max}, y_{max})$.



Algoritmus má inicializačnú fázu, testovaciu fázu a vlastný algoritmus je realizovaný v ....cykloch.

$P_1=(x_1, y_1)$, $P_2=(x_2, y_2)$

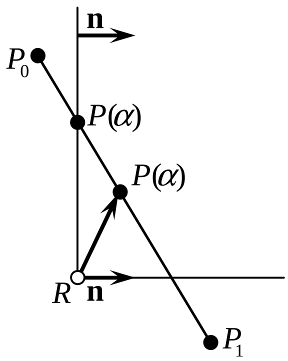
proc: <u>outcod</u> Ľ+P $y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$

proc: <u>swap</u> H+D $x = x_1 + \frac{x_2 - x_1}{y_2 - y_1}(y - y_1)$

fnc: <u>and</u> – log. s.

prem: <u>done</u> **HORE**
c. repeat until **DOLE**
bool.prem: <u>accept-</u> **VPRAVO**
inf . – vykr. **VĽAVO**

Orezávanie úsečky konvexným mnohouholníkom:LIANG-BARSKÉHO algoritmus.



- reprezentácia bodu na úsečke P_0P_1:

$$P(\alpha) = (1-\alpha)P_0 + \alpha P_1;\ 0 \le \alpha \le 1$$

- algoritmus počíta dve hodnoty parametra α_0, α_1, ktorým prislúcha výsledná orezaná úsečka $P(\alpha_0)P(\alpha_1)$ kde $\alpha_0 < \alpha_1$
- inicializácia: $\alpha_0 = 0$, $\alpha_1 = 1$ ($\Rightarrow$ t.j. $P(\alpha_0)P(\alpha_1) = P_0P_1$)
- prístup: orezávať úsečku vzhľadom na polroviny mnohouholníka
- skúmajme ako oreže úsečku polrovina $\langle R, \mathbf{n}\rangle$ kde **n** je normálový vektor hranice smerujúci dovnútra polroviny
- v priebehu algoritmu bude hodnota α_0 rásť (neklesať) a hodnota α_1 klesať (nerásť)
- ak $\alpha_0 > \alpha_1$, tak orezávaná úsečka je prázdna, algoritmus končí
- chceme teraz zistiť hodnotu α (ak $\exists$), pre ktorú priamka, na ktorej leží úsečka, pretína hranicu polroviny, presnejšie nájsť podmienku aby $P(\alpha) \in$ polrovine. Aby sme takéto α vypočítali, dosadíme $P(\alpha)$ do podmienky: patriť polrovine:

$$(P(\alpha) - R).\mathbf{n} \ge 0 \quad (P(\alpha) = (1-\alpha)P_0 + \alpha P_1 = P_0 + \alpha(P_1 - P_0))$$

a z nej vypočítame podmienku pre α:

$$(P_0 + \alpha(P_1 - P_0)) - R).\mathbf{n} \ge 0 \Leftrightarrow \alpha(P_1 - P_0).\mathbf{n} + (P_0 - R).\mathbf{n} \ge 0 \Leftrightarrow$$

$$\Leftrightarrow \alpha(P_1 - P_0).\mathbf{n} \ge (R - P_0).\mathbf{n} \Leftrightarrow \boxed{\alpha d_1 \ge d_r},$$

kde $d_1 = (P_1 - P_0).\mathbf{n}$ a $d_r = (R - P_0).\mathbf{n}$.

Potom existuje niekoľko prípadov v závislosti od hodnoty d_1:

1. $d_1 > 0$: Potom $\boxed{\alpha \ge d_r / d_1}$. Aby sme to zabezpečili stačí položiť: $\alpha_0 = \max(\alpha_0, d_r / d_1)$.

   Ak je $\alpha_0 > \alpha_1$ ako výsledok dostaneme, že orezávaná úsečka je prázdna –koniec.
2. $d_1 < 0$: Potom $\boxed{\alpha \le d_r / d_1}$. Aby sme to zabezpečili stačí položiť : $\alpha_1 = \min(\alpha_1, d_r / d_1)$.

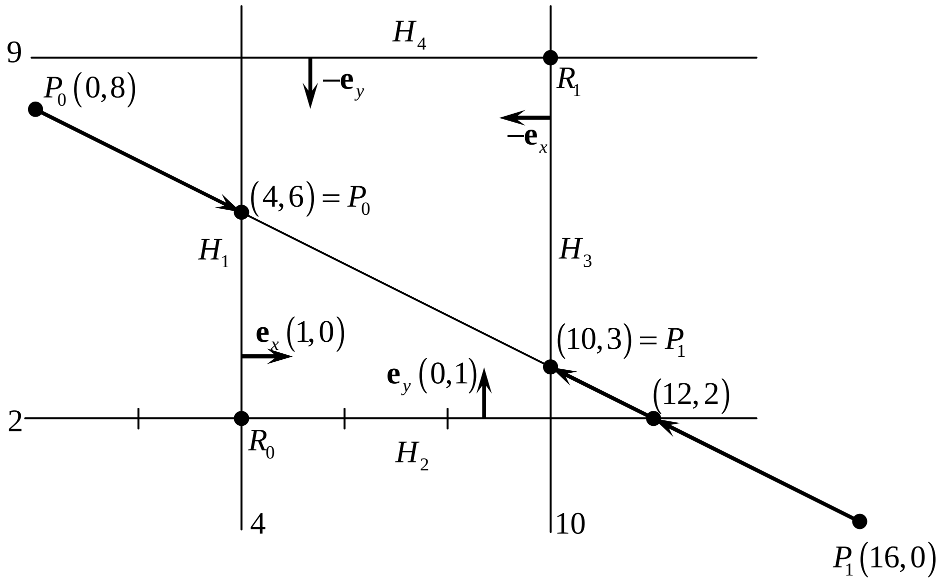
   Ak ako výsledok dostaneme $\alpha_0 > \alpha_1$, tak opäť orezávaná úsečka bude prázdna – koniec.
3. $d_1 = 0$: Potom α nie je definované. Geometricky to znamená, že hraničná priamka a orezávaná úsečka sú rovnobežné. V tomto prípade stačí vziať ľubovoľný bod na úsečke napr. P_0 a ak bude $(P_0 - R).\mathbf{n} < 0$, tak bude jasné, že celá úsečka je mimo polroviny a preto prienik s mnohouholníkom je prázdny. Inak neurobíme nič, čiže pokračujeme v orezávaní ďalšími polrovinami.

Príklad: Je dané okno: $4 \le x \le 10$ a $2 \le y \le 9$. Určenie polrovín si vyžaduje voľbu minimálne dvoch bodov, napr.: $R_0 = (4, 2, 1)^T$ a $R_1 = (10, 9, 1)^T$ a dvoch vektorov $\mathbf{e}_x = (1, 0, 0)^T$, $\mathbf{e}_y = (0, 1, 0)^T$. Pomocou nich môžeme vytvoriť nasledujúce štyri polroviny:

$H_1 = \langle R_0 = (4, 2, 1)^T, \mathbf{e}_x = (1, 0, 0)^T \rangle$; $H_2 = \langle R_0 = (4, 2, 1)^T, \mathbf{e}_y = (0, 1, 0)^T \rangle$;

$H_3 = \langle R_1 = (10, 9, 1)^T, -\mathbf{e}_x = (-1, 0, 0)^T \rangle$; $H_4 = \langle R_1 = (10, 9, 1)^T, -\mathbf{e}_y = (0, -1, 0)^T \rangle$.

Orežme podľa tohto algoritmu úsečku P_0P_1: $P_0 = (0, 8, 1)^T$; $P_1 = (16, 0, 1)^T$:



0°. Inicializácia: $\underline{\alpha_0 = 0,\ \alpha_1 = 1}$

1°. $H_1 = \langle R_0, \mathbf{e}_x \rangle$:

$$d_1 = (P_1 - P_0).\mathbf{e}_x = (16,-8).(1,0) = 16 > 0,$$

$$d_r = (R_0 - P_0).\mathbf{e}_x = (4,-6).(1,0) = 4 \Rightarrow \boxed{\alpha \ge \frac{1}{4}}$$

$$d_1 > 0:\ \alpha_0 = \max(0, 1/4) = \frac{1}{4};$$

1.čiastkový výsledok: $(\alpha_0 = \frac{1}{4}, \alpha_1 = 1)$

2°. $H_2 = \langle R_0, \mathbf{e}_y \rangle$:

$$d_1 = (P_1 - P_0).\mathbf{e}_x = (16,-8).(0,1) = -8 < 0; \quad d_r = (4,-6).(0,1) = -6;$$

$$d_1 < 0:\ \boxed{\alpha \le \frac{3}{4}} \Rightarrow \alpha_1 = \min(1, 3/4) = \frac{3}{4}$$

2.čiastkový výsledok: $(\alpha_0 = \frac{1}{4}, \alpha_1 = \frac{3}{4})$

3°. $H_3 = \langle R_1, -\mathbf{e}_x = (-1,0) \rangle$:

$$d_1 = (16,-8).(-1,0) = -16$$

$d_r = (R_1 - P_0).(-1,0) = (10,1).(-1,0) = -10$

$d_1 < 0$: $\boxed{\alpha \le \frac{5}{8}}$: $\alpha_1 = \min(3/4, 5/8) = \frac{5}{8}$

<u>3.čiastkový výsledok</u>: $(\alpha_0 = \frac{1}{4}, \alpha_1 = \frac{5}{8})$

4°. $H_4 = \langle R_1, -\mathbf{e}_y \rangle = \langle R_1 = (10, 9, 1)^T, -\mathbf{e}_y = (0, -1, 0)^T \rangle$

$d_1 = (16,-8).(-0,-1) = 8$

$d_r = (R_1 - P_0).(0,-1) = (10,1).(0,-1) = -1$

$d_1 > 0$: $\boxed{\alpha \ge -\frac{1}{8}}$: $\alpha_0 = \max(1/4, -1/8) = \frac{1}{4}$.

<u>Výsledok</u>: $(\alpha_0 = \frac{1}{4},\ \alpha_1 = \frac{5}{8})$. Teda krajné body orezanej úsečky sú:

- pre $\alpha_0 = \frac{1}{4}$: $\frac{3}{4}(0,8,1) + \frac{1}{4}(16,0,1) = (4,6,1) = P_0$
- pre $\alpha_1 = \frac{5}{8}$: $\frac{3}{8}(0,8,1) + \frac{5}{8}(16,0,1) = (10,3,1) = P_1$.

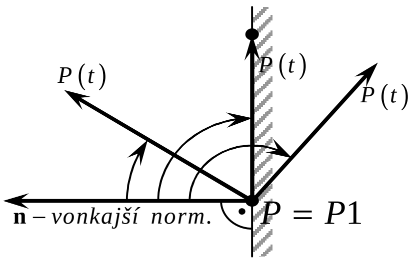
Orezávací algoritmus Cyrus-Beck (pre mnohouholníky)

Staršia verzia Liang-Barského algoritmu je známa ako algoritmus Cyrus-Beck-a. Je založený na hľadaní tzv. vstupného bodu úsečky $P1P2$ do okna, ktorý je určený maximálnym vstupným parametrom t_e a výstupného bodu úsečky $P1P2$ z okna, ktorý je určený zasa minimálnym výstupným parametrom t_o v intervale <0,1>.

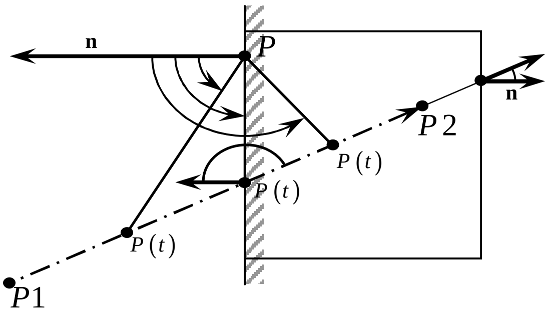
Predpokladá sa, že úsečka $P1P2$ je určená parametrickým vyjadrením:

$$P(t) = P1 + \mathbf{d}t \text{, kde } \mathbf{d} = P2 - P1 \text{ a } t \in \langle 0,1 \rangle$$

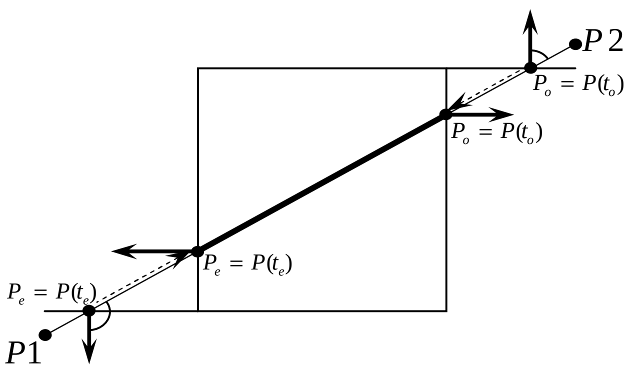
a chceme určiť jej prienik so stranou E (edge) konvexného mnohouholníka určenou bodom P a vonkajším normálovým vektorom $\mathbf{n}$.



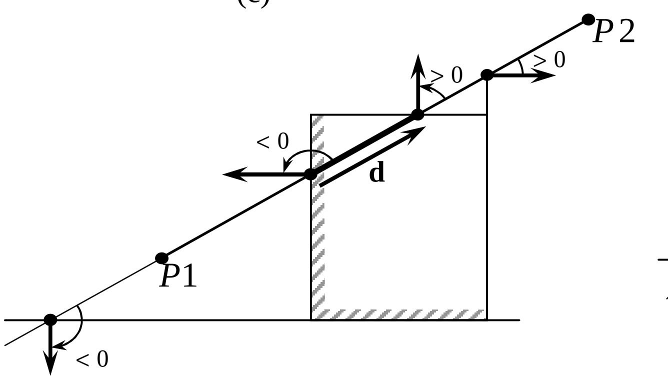
(a)



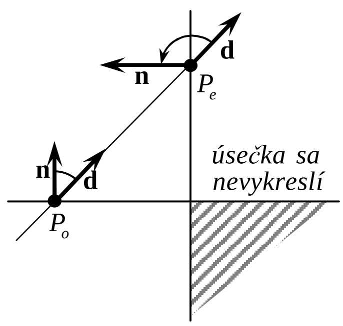
(b)



(c)



(d)



(e)

Poznámka: z obrázkov (c), (d) a (e) vyplýva: pre body P_e je charakteristické, že $\mathbf{n}.\mathbf{d} \le 0$ a pre body P_o, že $\mathbf{n}.\mathbf{d} \ge 0$.
Potom:

1. $P(t)$ je vonkajším bodom $\Leftrightarrow \sphericalangle[P(t)-P,\mathbf{n}] \in \langle 0, \pi/2)$ $\Leftrightarrow$ $[P(t)-P].\mathbf{n} > 0$
2. $P(t)$ je vnútorným bodom $\Leftrightarrow \sphericalangle[P(t)-P,\mathbf{n}] > \pi/2$ $\Leftrightarrow$ $[P(t)-P].\mathbf{n} < 0$
3. $P(t) \in$ strany $E \Leftrightarrow [P(t)-P].\mathbf{n} = 0 \Leftrightarrow (P1 - P + t\mathbf{d}).\mathbf{n} = 0 \Leftrightarrow (P1-P).\mathbf{n} + t(\mathbf{d}.\mathbf{n}) = 0 \Leftrightarrow$

$$\Leftrightarrow \left[\mathbf{d}.\mathbf{n} \ne 0 \wedge t = -\frac{(P1-P).\mathbf{n}}{\mathbf{d}.\mathbf{n}}\right] \vee \left[\mathbf{d}.\mathbf{n} = 0 \wedge (P1-P).\mathbf{n} = 0\right] \vee \left[\mathbf{d}.\mathbf{n} = 0 \wedge (P1-P).\mathbf{n} \ne 0\right] \Leftrightarrow$$

$$t = -\frac{(P1-P).\mathbf{n}}{\mathbf{d}.\mathbf{n}} \vee$$

$\vee$ priamka $P1P2$ obsahuje stranu mnohouholníka t.j. prienikom priamky a mnohouholníka je táto strana, alebo strana je orezaním priamky $P1P2$ mnohouholníkom – koniec $\vee$
$\vee$ priamka $P1P2$ a strana E nemajú spoločný ani jeden bod, čiže priamka je so stranou E rovnobežná, pričom môže, ale nemusí pretínať ďalšie strany – neurobíme nič (presnejšie, prejdeme k ďalšej strane ak to bude aktuálne).

V tomto 3. prípade nám pre pokračovanie zostala jediná možnosť:
$P(t) \in$ strany $E \Leftrightarrow t = -\frac{(P1-P).\mathbf{n}}{\mathbf{d}.\mathbf{n}}$, kde $\mathbf{d}.\mathbf{n} \ne 0$, čiže môže byť $\mathbf{d}.\mathbf{n} < 0$ alebo $\mathbf{d}.\mathbf{n} > 0$.

Ak vo výraze $-\frac{(P1-P).\mathbf{n}}{\mathbf{d}.\mathbf{n}}$ je $\mathbf{d}.\mathbf{n} < 0$ [$\mathbf{d}.\mathbf{n} > 0$] nazveme ho potenciálne vstupným [výstupným] parametrom a označíme t_e [t_o] a bod $P(t_e)$ [$P(t_o)$] potenciálne vstupným [výstupným] bodom úsečky $P1P2$ do okna [z okna].

V priebehu algoritmu postupne položíme $E = E_i$, $\mathbf{n} = \mathbf{n}_i$, $P = P_i$, pre i=1,2,3,4.

Vlastný algoritmus C-B.

Procedure Cyrus-Beck

begin
1. if (P1=P2) then clip_point {úsečka=bod, orež bod}
   else
   begin
2. t_e=0 {inicializácia parametrov}
   t_o=1
   d=P2-P1 {pre ∀ hranu okna a jej vonkajší normálový vektor}
3. for $E_i \wedge \mathbf{n}_i$ (i=1,2,3,4)
   begin $\mathbf{n}.\mathbf{d} = \mathbf{n}_i .\mathbf{d}_i$
4. if $\mathbf{n}.\mathbf{d} \ne 0$ then t= - $\mathbf{n}.(P1-P_i)/\mathbf{n}.\mathbf{d}$; {výpočet parametra}
   if ($\mathbf{n}.\mathbf{d} < 0$) $\wedge$($t \le 1$) then t_e =max(t_e, t) {úprava parametra potenc. vstupu}
   if ($\mathbf{n}.\mathbf{d} > 0$) $\wedge$($t \ge 0$) then t_o =min(t_o, t) {úprava parametra potenc. vstupu}
   end
5. if $t_e > t_o$ then return nil ; {úsečka mimo okna}
   else

```
    return (P(t_e), P(t_o));                          {výstup- orezaná úsečka}
    end
end
```

Poznámka: Pozor na možné chyby – nekriticky prevzaté z literatúry. Preveriť a opraviť na príklade k algoritmu Liang-Barsky.