GEOMETRIAI FELADATOK
(ESETTANULMÁNY)

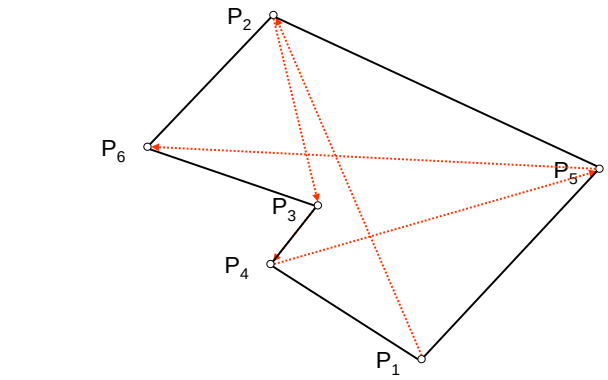
TARTALOM

Az esettanulmány Horváth Gyula: „*Geometriai algoritmusok*” c., az NJSzT által gondozott „Tehetséggondozó Program” keretén belül megjelent kötete alapján készült.

BEVEZETÉS

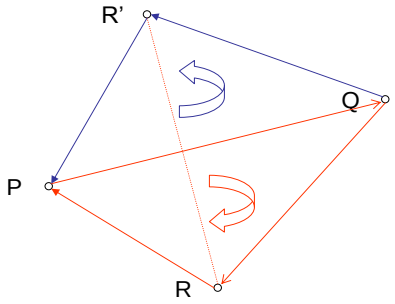
A feladatok köre…

- Pontok összekötése zárt, nem-metsző poligonná.



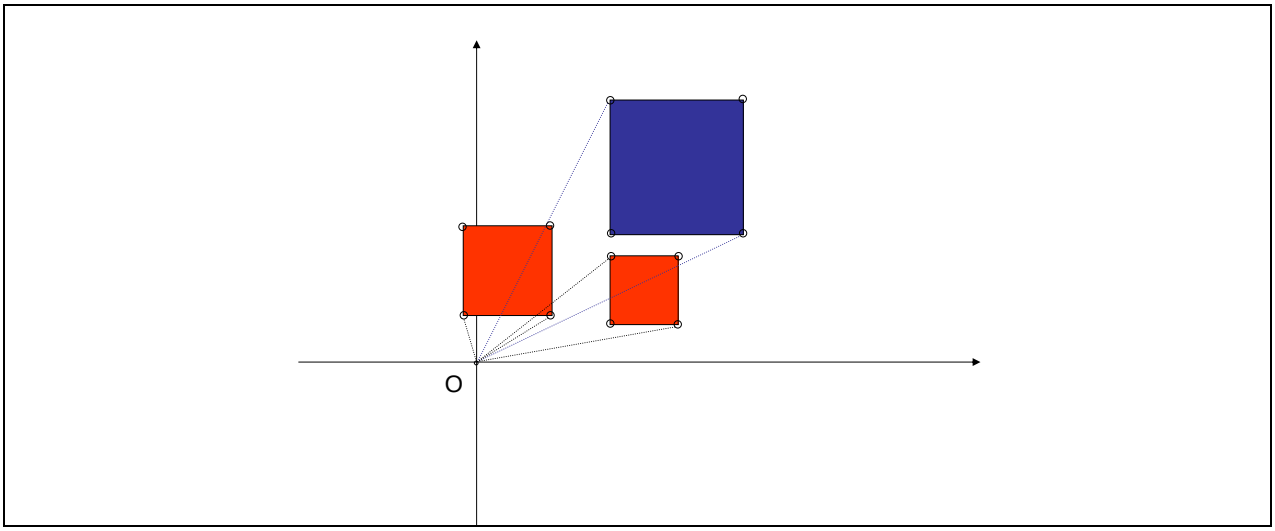
$P_1 \rightarrow P_5 \rightarrow P_2 \rightarrow P_6 \rightarrow P_3 \rightarrow P_4$

- 3 pont „forgásiránya”.



- Egy pont adott poligon belső pontja-e?
- Egy pont adott szakaszra illeszkedik-e?
- 2 szakasz metsző-e? Ha igen, mi a metszéspontjuk?
- Ponthalmaz konvex burka.

- A(z origóból) látható négyzetek (pl. megszámlálása).



A fentiek megoldásában szereplő alapvető típusok (ábrázolás+művelethalmaz): pont, szakasz, pontsorozat.

Pont

- Ábrázolás:

```
TPont=Rekord(x,y:Valós)
```

- Asszociált műveletek szignatúrája:

```
WritelnTPont(Konst p:TPont)
```

Problémamentes.

```
ReadlnTPont(Vált p:TPont)
```

Problémamentes.

```
ForgásIrány(Konst p,q,r:TPont):{-1,0,+1}
[p→q→r balforgás[1], kollineáris[2], jobbforgású esetben]
```

Def-x:

$\_ \times \_ : TPont^2 \to R$

$p_1 \times p_2 := p_1.x * p_2.y - p_2.x * p_1.y$

Megjegyzés:

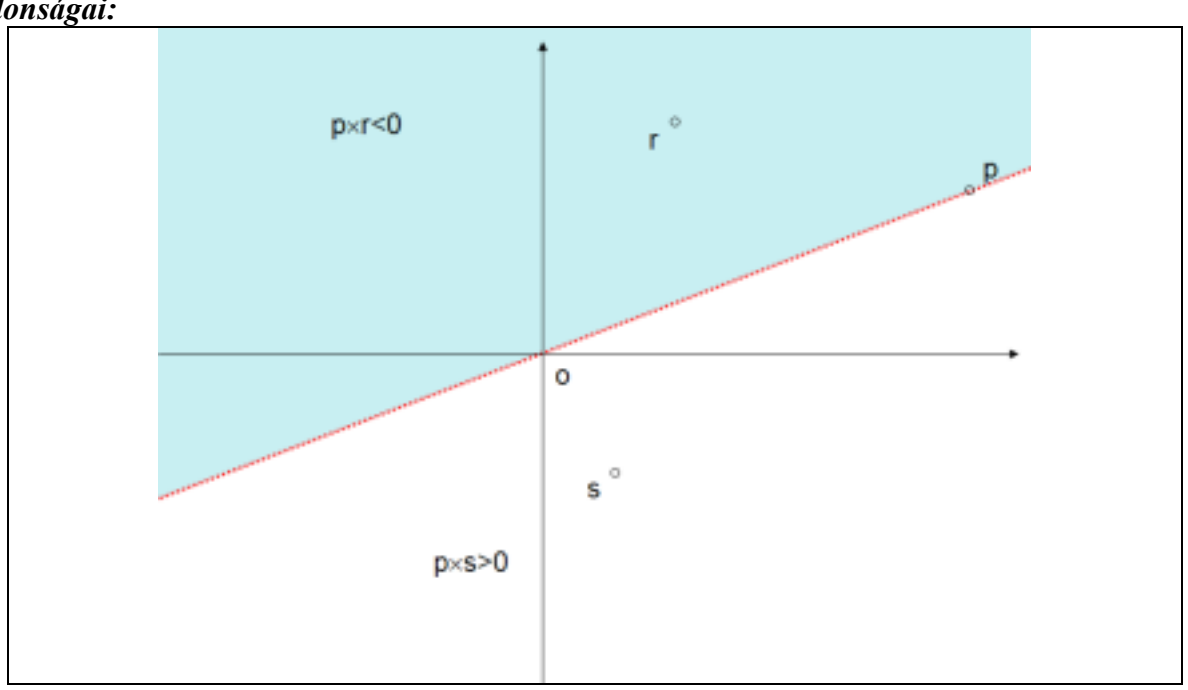
A $p_1 \times p_2$ művelet a ($\varnothing, p_1, p_2, p_1+p_2$) pontok által kijelölt paralelogramma előjeles területét adja.

[1] Azaz az óra járásával ellentétes irányúak.

[2] Azaz egy egyenesre illeszkednek.

Tulajdonságai:

(T0)



Ha a p az 1. vagy a 4. síknegyedben van, akkor az op-re illeszkedő egyenes feletti pontokra a p×r>0 alatti pontokra >0.

Biz.: $p\times r=p.x*r.y-r.x*p.y>0 \Rightarrow p.x*r.y>r.x*p.y$

$ha\ p.x>0,\ akkor\ r.y>r.x*p.y/p.x,\ azaz\ r.y>r.x*iránytangens(p)$

$ha\ p.x<0,\ akkor\ r.y<r.x*p.y/p.x,\ azaz\ r.y<r.x*iránytangens(p)$

♦

(T1) $p\times p=0$

Biz.: $p\times p=$

$=p.x*p.y-p.x*p.y=0$

♦

(T2) $p_1\times p_2=-p_2\times p_1$

Biz.: $p_1\times p_2=p_1.x*p_2.y-p_2.x*p_1.y=$

$=-(-p_1.x*p_2.y+p_2.x*p_1.y)=-p_2\times p_1$

♦

(T3) $(a*p_1)\times(b*p_2)=a*b*(p_1\times p_2)$

Biz.: $(a*p_1)\times(b*p_2)=(a*p_1.x)*(b*p_2.y)-(b*p_2.x)*(a*p_1.y)=$

$=a*b*(p_1.x*p_2.y-p_2.x*p_1.y)=a*b*(p_2\times p_1)$

♦

(T4) $(p_1+q)\times p_2=p_1\times p_2+q\times p_2$

Biz.: $(p_1+q)\times p_2=(p_1.x+q.x)*p_2.y-p_2.x*(p_1.y+q.y)=$

$=p_1.x*p_2.y+q.x*p_2.y-p_2.y*p_1.y-p_2.x*q.y=$

$=p_1\times p_2+q\times p_2$

♦

(T5) $p_1 \times (p_2+q) = p_1 \times p_2 + p_2 \times q$

Biz.: $p_1 \times (p_2+q) =$

$(T2) \Rightarrow \quad = -(p_2+q) \times p_1 =$

$(T3) \Rightarrow \quad = -p_2 \times p_1 - q \times p_1 =$

$(T2) \Rightarrow \quad = p_1 \times p_2 + p_1 \times q$

♦

Megjegyzés:

(T4)&(T5) ⇒ a × disztributív a + műveletre nézve

(T3)&(T4)&(T5) ⇒ a × a vektortéren linearitást tartó leképezés

Def-KSz:

$KeresztSzorzat: TPont^3 \rightarrow R$

$KeresztSzorzat(p,q,r) := (q-p) \times (r-p) =$

$= (q.y-p.y)*(r.x-p.x) - (r.y-p.y)*(q.x-p.x)$

Állítás:

Ha a KeresztSzorzat(p,q,r)>0 , akkor a KeresztSzorzat(p,q,r')<0, ahol r' az r tükörképe a p_q-ra illeszkedő egyenesre nézve.

Bizonyítás:

(Def-KSz) ⇒ KeresztSzorzat(p,q,r)=(q-p)×(r-p)

Az világos, hogy

az r a p_q-ra illeszkedő egyenes alatt van ⇔

ha r-p a 0_q-p-re illeszkedő egyenes alatt van

(T0) ⇒ (q-p)×(r-p)>0 ⇔

ha r-p a 0_q-p-re illeszkedő egyenes alatt van

No már most az r' éppen úgy helyezkedik el a p_q egyeneshez képest, mint az r'-p a 0_q-p egyeneshez képest: azaz mindkét esetben a megfelelő egyenes felett lesz.

(T0) ⇒ (q-p)×(r'-p)<0 ⇒

(q-p)×(r'-p)=KeresztSzorzat(p,q,r')<0

♦

Az állítás következménye, hogy a `ForgásIrány`-számítást alapozni lehet a `Kereszt-Szorzat` képletére.

○ Szakasz

- Ábrázolás:

```
TSzakasz=Rekord(p,q:TPont)
```

- Asszociált műveletek szignatúrája:

```
WritelnTSzakasz(Konst s:TSzakasz)
```

Problémamentes. l. a programban (55. sor környékén).

```
ReadlnTSzakasz(Vált s:TSzakasz)
```

Problémamentes; l. a programban (60. sor környékén).

```
SzakaszonE(Konst s:TSzakasz; r:TPont):Logikai
```

Állítás:

p,q,r∈TPont egy egyenesen vannak ⇔ KeresztSzorzat(p,q,r)=0

Bizonyítás:

(⇒)

p,q,r∈TPont egy egyenesen vannak ⇒ ∃λ∈R: r=p+λ(q-p) ⇒*

KeresztSzorzat(p,q,r)=(q-p)×(r-p)=(q-p)×(p+λ(q-p)-p)=*

=(q-p)×(λ(q-p))=*

(T3) ⇒ =λ(q-p)×(q-p)=*

(T1) ⇒ =λ(q-p)×(q-p)=0*

(⇐)

KeresztSzorzat(p,q,r)=0 ⇒

(q.y-p.y)(r.x-p.x)-(r.y-p.y)*(q.x-p.x)=0 ⇒*

(q.y-p.y)(r.x-p.x)=(r.y-p.y)*(q.x-p.x) ⇒*

1. ha (q.x-p.x)≠0 és (r.x-p.x)≠0 ⇒

(q.y-p.y)/(q.x-p.x)=(r.y-p.y)/(r.x-p.x) ⇒

Iránytangens(q-r)=Iránytangens(r-p) ⇒

p,q,r∈TPont egy egyenesen vannak.

2. ha (q.x-p.x)=0 ⇒

p_q-n átmenő egyenes az x-tengelyre merőleges és

(r.x-p.x)=0 vagy (q.y-p.y)=0 ⇒

2a.ha (r.x-p.x)=0 ⇒

p_r-n átmenő egyenes az x-tengelyre merőleges ⇒

p,q,r∈TPont egy egyenesen vannak.

2b. ha (q.y-p.y)=0 ⇒

p=q ⇒

p,q,r∈TPont egy egyenesen vannak.

♦

Állítás:

ha p,q,r∈TPont egy egyenesen vannak és

r.x∈[Min(p.x,q.x)..Max(p.x,q.x)],

akkor az r a p_q szakaszon

Bizonyítás:

Nyilvánvaló.

♦

Ezen állításokból adódik már a függvény algoritmusa. L. hátrébb a 70. sor környékén.

```
SzakaszPárMetszőE(Konst s1,s2:TSzakasz):Logikai
```

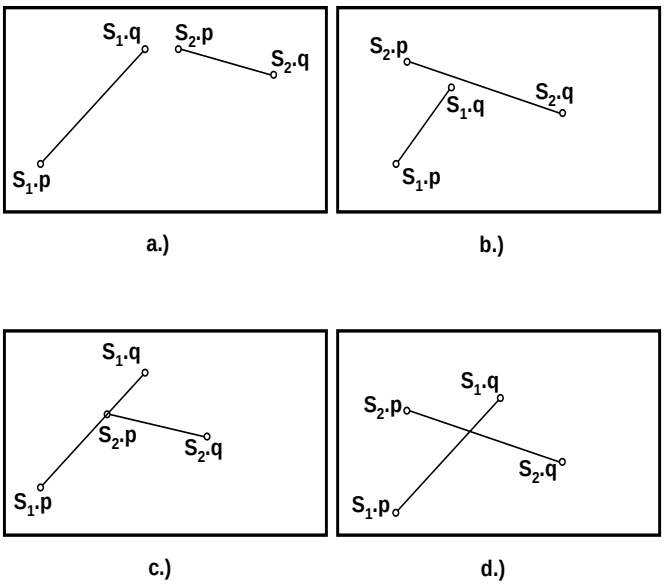
Állítás:

A szakaszok metszőség-vizsgálatát a ForgásIrány-vizsgálatra vissza lehet vezetni.

Bizonyítás:

Az alábbi alapesetek képzelhetők el:

Alapesetek:



a.) $ForgásIrány(s_1.p,s_1.q,s_2.p)=ForgásIrány(s_1.p,s_1.q,s_2.q)$ és
$ForgásIrány(s_2.p,s_2.q,s_1.p)=ForgásIrány(s_2.p,s_2.q,s_1.q)$

b.) $ForgásIrány(s_2.p,s_2.q,s_1.p)=ForgásIrány(s_2.p,s_2.q,s_1.q)$ és
$ForgásIrány(s_1.p,s_1.q,s_2.p)=-ForgásIrány(s_1.p,s_1.q,s_2.q)$

c.) $ForgásIrány(s_1.p,s_1.q,s_2.p)=0$ és $ForgásIrány(s_1.p,s_1.q,s_2.q)\neq 0$ és
$ForgásIrány(s_2.p,s_2.q,s_1.p)=-ForgásIrány(s_2.p,s_2.q,s_1.q)$

d.) $ForgásIrány(s_2.p,s_2.q,s_1.p)=-ForgásIrány(s_2.p,s_2.q,s_1.q)$ és
$ForgásIrány(s_2.p,s_2.q,s_1.p)=-ForgásIrány(s_2.p,s_2.q,s_1.q)$

♦

Ezen állításból adódik már a függvény algoritmusa. L. a programban a 80. sor környékén!

```
SzakaszPárMetszéspont(Konst s1,s2:TSzakasz; Vált r:TPont)
```

Állítás:

Az s_1 s_2 szakaszoknak az r metszéspontja, ha van metszéspontja s_1-nek és s_2-nek, továbbá
$\exists t_i\in[0..1]: r_i=s_i.p+t_i*(s_i.q-s_i.p)$ $(i=1,2)$, ekkor $r:=r_1=r_2$

Bizonyítás:

Tfh. Létezik közös pont.

Az alábbi egyenletek megoldása szolgáltatja a megoldást:

($r_1=r_2=r$)

$s_1.p+t_1*(s_1.q-s_1.p)=s_2.p+t_2*(s_2.q-s_2.p)$ mind x-, mind y-koordinátára ⇒

… ennek megoldása t_i-kre, majd (a közös és keresett) r.

♦

Ezen állításból adódik már a függvény algoritmusa. L. a programban a 101. sor környékén)!

○ Pontsorozat

- Ábrázolás:

  TPontok=**Tömb**(1..MaxN:TPont)

- Asszociált műveletek szignatúrája:

  …

A geometriai típusok egyesített modulja (SZAKPONT.INC):

```
1   (*
2     A szakasz és pont típusának megvalósítása.
3     Export:
4       Konst MaxN:Egész
5       Típus TPont=Rekord(x,y:Valós)
6             Eljárás WritelnTPont(Konst p:TPont)
7                     ReadlnTPont(Vált p:TPont)
8                     ForgásIrány(Konst p,q,r:TPont)
9       Típus TPontok=Tömb(1..MaxN:TPont)
10      Típus TSzakasz=Rekord(p,q:TPont)
11            Eljárás  WritelnTSzakasz(Konst s:TSzakasz)
12                     ReadlnTSzakasz(Vált s:TSzakasz)
13            Függvény SzakaszonE(Konst s:TSzakasz; r:TPont):Logikai
14                     SzakaszPárMetszőE(Konst s1,s2:TSzakasz):Logikai
15            Eljárás  SzakaszPárMetszéspont(Konst s1,s2:TSzakasz; Vált r:TPont)
16  *)
17    Const
18      MaxN = 9;
19    Type
20      TPont = Record x,y:Real End;
21      Procedure WritelnTPont(Const p:TPont);
22      Begin
23        Writeln(' x:',p.x:6:3,' , y:',p.y:6:3);
24      End;{WritelnTPont}
25      Function ReadlnTPont(Var p:TPont):Boolean;
26      Begin
27        {$i-}
28        Write('x-,y-koordináták:');
29        Readln(p.x,p.y);
30        {$i+}
31        ReadlnTPont:=IOResult=0
32      End;{ReadlnTPont}
33      Function ForgasIrany(Const p,q,r:TPont):Integer;
34      (*
35        Uf: p->q->r jobbforgású => ForgasIrany(p,q,r)=+1
36            p->q->r balforgású  => ForgasIrany(p,q,r)=-1
37            p-q-r kollineárisak => ForgasIrany(p,q,r)=0
38      *)
```

```
39        Var
40          keresztSzorzat:Real;
41      Begin
42        keresztSzorzat:=(q.y-p.y)*(r.x-p.x)-(r.y-p.y)*(q.x-p.x);
43        If
44          keresztSzorzat<0 then ForgasIrany:=-1 else if
45          keresztSzorzat>0 then ForgasIrany:=+1
46                           else ForgasIrany:=0
47        {EndIf}
48      End;{ForgasIrany}
49
50    Type
51      TPontok = Array [1..MaxN] of TPont;
52
53    Type
54      TSzakasz = Record p,q:TPont End;
55      Procedure WritelnTSzakasz(Const s:TSzakasz);
56      Begin
57        Write('Kezdőpont: '); WritelnTPont(s.p);
58        Write('Végpont: ');   WritelnTPont(s.q);
59      End;{WritelnTSzakasz}
60      Function ReadlnTSzakasz(Var s:TSzakasz):Boolean;
61      Begin
62        Writeln('Adja meg a kezdő- és végpontot:');
63        ReadlnTSzakasz:=ReadlnTPont(s.p) and ReadlnTPont(s.q);
64      End;{ReadlnTSzakasz}
65      Function SzakaszonE(Const s:TSzakasz; Const r:TPont):Boolean;
66      (*
67        Uf: SzakaszonE(s,r)=az s szakaszra illeszkedik-e az r pont
68      *)
69      Begin
70        SzakaszonE:={az s EGYENESére illeszkedik-e}
71               ((r.x-s.p.x)*(s.q.y-s.p.y)=(r.y-s.p.y)*(s.q.x-s.p.x)) and
72               {az s SZAKASZra illeszkedik-e}
73               (Min(s.p.x,s.qx)<=r.x) and (r.x<=Max(s.p.x,s.q.x))
74      End;{Szakaszon}
75
76      Function SzakaszParMetszoE(Const s1,s2:TSzakasz):Boolean;
77        Var
78          fpq1,fpq2,fqp1,fqp2:Integer{-1,0,+1};
79      Begin
80        fpq1:=ForgasIrany(s1.p,s1.q,s2.p);
81        fpq2:=ForgasIrany(s1.p,s1.q,s2.q);
82        fqp1:=ForgasIrany(s2.p,s2.q,s1.p);
83        fqp2:=ForgasIrany(s2.p,s2.q,s1.q);
84        SzakaszParMetszoE:=(fpq1*fpq2<0) and (fqp1*fqp2<0) or
85                           SzakaszonE(s1,s2.p) or
86                           SzakaszonE(s1,s2.q) or
87                           SzakaszonE(s2,s1.p) or
88                           SzakaszonE(s2,s1.q)
89      End;{SzakaszParMetszoE}
90
91      Procedure SzakaszParMetszespont(Const s1,s2:TSzakasz; Var r:TPont);
92      (*
93        Ef: SzakaszParMetszoE(s1,s2) -- s1, s2 egymást metsző szakaszok
94        Uf: r rajta van az s1-n és az s2 is
95      *)
96        Var
97         ax,aax,bx,bbx,
98         ay,aay,by,bby,
99         t:Real;
```

```
100       Begin
101         ax:=s1.p.x; bx:=s1.q.x-s1.p.x; aax:=s2.p.x; bbx:=s2.q.x-s2.p.x;
102         ay:=s1.p.y; by:=s1.q.y-s1.p.y; aay:=s2.p.y; bby:=s2.q.y-s2.p.y;
103         t:=((ay-aay)*bx+(aax-ax)*by)/(bby*bx-bbx*by);
104         r.x:=s2.p.x+t*(s2.q.x-s2.p.x);
105         r.y:=s2.p.y+t*(s2.q.y-s2.p.y);
106       End;{SzakaszParMetszespont}
```

1. FELADAT

A fenti típusok műveleteinek gyakorlásaként adjuk meg két szakasz metszéspontját, ha van!

1. FELADAT – MEGOLDÁS

Demó. (METSZES.EXE)

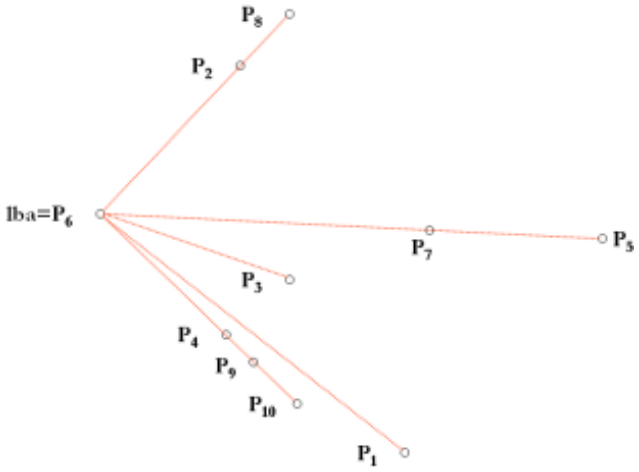
```
1   Program MetszoSzakaszok; {SzP 05.02.11.}
2   (*
3     Geometriai feladat 1.
4     Feladat: Adott szakaszpár metszés-vizsgálata.
5   *)
6     Uses
7       Newdelay,Crt22,Crt;
8     {$i AltRutin.inc}{$i SzakPont.inc}
9     Const
10      foCim='Metsző szakaszok';
11    Var
12      s1,s2:TSzakasz;
13      r:TPont;
14
15  Begin
16    UjLap(foCim+' -- beolvasás',0);
17    Repeat
18      Writeln('Adja meg az S1 szakaszt!');
19    Until ReadlnTSzakasz(s1);
20    Repeat
21      Writeln('Adja meg az S2 szakaszt!');
22    Until ReadlnTSzakasz(s2);
23    UjLap(foCim+' -- feldolgozás',-1);
24    If SzakaszParMetszoE(s1,s2) then
25    Begin
26      Writeln('Van metszéspontjuk, mégpedig:');
27      SzakaszParMetszespont(s1,s2,r);
28      WritelnTPont(r);
29    End
30       else
31    Begin
32      Writeln('Nincs metszéspontjuk.')
33    End;
34    BillreVar;
35  End.
```

2. FELADAT

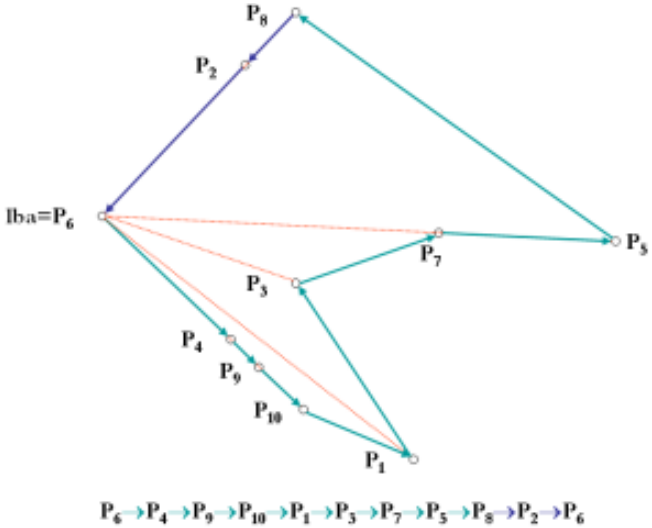
Adott N darab (nem kollineáris) pont. Adjuk meg a pontok olyan sorrendjét, amelyben az egymást követőket, és az utolsót az elsővel összekötve zárt, nem-metsző poligont kapunk.

2. FELADAT – A-MEGOLDÁS

Arra építjük a megoldást, hogy a ponthalmaz pontjainak egy alkalmas sorrendjét kapjuk, ha a legbaloldalibb-legalsó pontból „nézve” iránytangensük szerint rendezzük.



Példa. A ponthalmaz és lba-ból kiinduló és az egyes pontokban végződő egyenesek.



Példa. A pontok és iránytangensük (valamint lba-hoz való közelségük) szerinti sorrendje,

Demó. (PONTSORR.EXE)

```
1   Program PontSorrend; {SzP 05.02.11.}
2   (*
3     Geometriai feladat 2.
4     Feladat: adott N (nem kollineáris) pont. Adjuk meg a pontok
5              olyan sorrendjét, amelyben az egymást követőket,
6              és az utolsót az elsővel összekötve zárt, nem-metsző
7              poligont kapunk.
8   *)
9     Uses
10      Newdelay,Crt22,Crt;
11    {$i AltRutin.inc}{$i SzakPont.inc}
12    Const
13      foCim='N ponton nyugvó zárt poligon';
14    Var
15      p,pp:TPontok;
16      s:TSzakasz;
17      N,i,k:Integer;
18
```

```
19    Function EgyEgyenesenE(Const N:Integer; Const p:TPontok):Boolean;
20    (*
21      Uf: EgyEgyenesenE(N,p)=BÁRMELY i ELEME [3..N] :
22                             LÉTEZIK t ELEME Valós : p[i]=p[1]+t*(p[2]-p[1])
23    *)
24      Var
25        i,j:Integer;
26        rajtaVan:Boolean;
27        t:Real;
28    Begin
29      {p[j:2..N] pont keresése: p[1].x<>p[j].x}
30      j:=2;
31      While (j<=N) and (p[j].x=p[1].x) do
32        Inc(j);
33      If j<=N then
34      Begin {nem mind van a p[1] "fölött", pl. a j. ilyen}
35        If j>2 then {garantáltan nem esnek egy egyenesre, hiszen
36                     a j-1.-ig p[1]-en átmenő függőlegesre, és
37                     a j. egy "ferde" egyenesre esik}
38        Begin
39          EgyEgyenesenE:=False
40        End
41          else {j=2, lehet, hogy egy "ferde" egyenesre esik mind}
42        Begin
43          {p[1]->p[2] iránytangense:}
44          t:=(p[2].y-p[1].y)/(p[2].x-p[1].x);
45          rajtaVan:=True;
46          i:=2;
47          While (i<N) and rajtaVan {az i. a p[1]->p[2] egyenesen nyugszik} do
48          Begin
49            Inc(i);
50            rajtaVan:=p[i].y=p[1].y+(p[i].x-p[1].x)*t
51          End;
52          EgyEgyenesenE:=rajtaVan
53        End{If j in}
54      End
55        else
56      Begin {egyetlen függőleges egyenesen fekszenek}
57        EgyEgyenesenE:=True
58      End{If j<=N}
59    End;{EgyEgyenesenE}
60
61    Procedure Poligon(Const N:Integer; Var p:TPontok;
62                                       Var pp:TPontok);
63    (*
64      Uf: a feladat meghatározása szerinti sorrendben írja ki a pontokat
65    *)
66      Var
67        i,k:Integer;
68
69      Procedure PolarSzogSzerintRendez(Const N:Integer; Var p:TPontok);
70      (*
71        Uf: BÁRMELY i ELEME [1..N-1] : fi(q,p[i])<fi(q,p[i+1])  ÉS
72            q ELEME p[1..N] ÉS
73            BÁRMELY i ELEME [1..N] : q.x<=p[i].x VAGY (q.x=p[i].x => q.y<=p[i].y)
74        Def: fi: TPont x TPont -> Valós
75             fi(q,p):=arctg((p.y-q.y)/(p.x-q.x))
```

```
76        Megj.: az algoritmus működik az arctg nélkül is, hiszen
77            rendezéshez fi helyett tg(fi) is jó, azaz a rendezés a
78            ((p[i].y-q.y)/(p[i].x-q.x))<((p[i+1].y-q.y)/(p[i+1].x-q.x))
79            alapján is elvégezhető
80      *)
81        Type
82          TIrTg=Record vegtelen:Boolean; irTg:Real End;
83        Var
84          i,j,
85          mini,legbal:Integer;
86          q:TPont;
87          plSzog: Array [1..MaxN] of TIrTg;
88          ir:TIrTg;
89      Begin
90        {legbal-alul levő kikeresése:}
91        legbal:=1;
92        For i:=2 to N do
93        Begin
94          If (p[legbal].x>p[i].x) or
95             (p[legbal].x=p[i].x) and (p[legbal].y>p[i].y) then legbal:=i
96        End;{For i}
97        {polárszög-számítás:}
98        For i:=1 to N do
99        Begin
100         If (p[legbal].x=p[i].x) then
101         Begin
102           polSzog[i].vegtelen:=True;
103           polSzog[i].irTg:=0;
104         End
105           Else
106         Begin
107           polSzog[i].vegtelen:=False;
108           polSzog[i].irTg:=(p[legbal].y-p[i].y)/(p[legbal].x-p[i].x);
109         End;{If (p[legbal].x}
110       End;{For i}
111       {legbal-alul levőnek az 1. helyre tétele:}
112       q:=p[legbal]; p[legbal]:=p[1]; p[1]:=q;
113       ir:=polSzog[legbal]; polSzog[legbal]:=polSzog[1]; polSzog[1]:=ir;
114       {a többi polárszög szerinti rendezése:}
115       For i:=2 to N-1 do
116       Begin
117         mini:=i;
118         For j:=i+1 to N do
119         Begin
120           If not polSzog[j].vegtelen and
121                  ((polSzog[mini].vegtelen) or
122                   (polSzog[mini].irTg>polSzog[j].irTg)) or
123                  ((not polSzog[mini].vegtelen) and
124                   (polSzog[mini].irTg=polSzog[j].irTg) and
125                   (p[mini].x>p[j].x))
126              {p[j]>p[mini]} then mini:=j
127         End;{For j}
128         q:=p[i]; p[i]:=p[mini]; p[mini]:=q;
129         ir:=polSzog[i]; polSzog[i]:=polSzog[mini]; polSzog[mini]:=ir;
130       End;{For i}
131     End;{PolarSzogSzerintRendez}
132
```

```
133   Begin{Poligon}
134     PolarSzogSzerintRendez(N,p);
135     {sorrend-korrekció -
136      az utolsó néhány egy egyenesen nyugvó sorrendje fordított:}
137     i:=N-1;
138     While (p[n].y-p[1].y)*(p[i].x-p[1].x)=
139           (p[i].y-p[1].y)*(p[n].x-p[1].x) do Dec(i);
140     Writeln('Az alábbi sorrendben kell a pontokat összekötni:');
141     db:=0;
142     {a fordítottak:}
143     For k:=1 to i do
144     Begin
145       WritelnTPont(p[k]); Inc(db); pp[db]:=p[k];
146     End;
147     {a "többiek":}
148     For k:=n downto i+1 do
149     Begin
150       WritelnTPont(p[k]); Inc(db); pp[db]:=p[k];
151     End;
152     WritelnTPont(p[1]); Inc(db); pp[db]:=p[1];
153   End;{Poligon}
154
155 Begin
156   UjLap(foCim+' -- beolvasás',0);
157   Repeat
158     Writeln('Adja meg a pontok számát!');
159     {$i-}
160     Readln(N);
161     {$i+}
162   Until (IOResult=0) and (N in [0..MaxN]);
163   For i:=1 to N do
164   Begin
165     Repeat
166       Writeln('Adja meg a(z) ',i,'. pontot!');
167     Until ReadlnTPont(p[i]);
168   End;
169   UjLap(foCim+' -- feldolgozás',-1);
170   If EgyEgyenesenE(N,p) then
171   Begin
172     Writeln('Egy egyenesen fekszenek. Nincs megoldás.')
173   End
174     else
175   Begin
176     Poligon(N,p);
177   End;
178   BillreVar;
179 End.
```

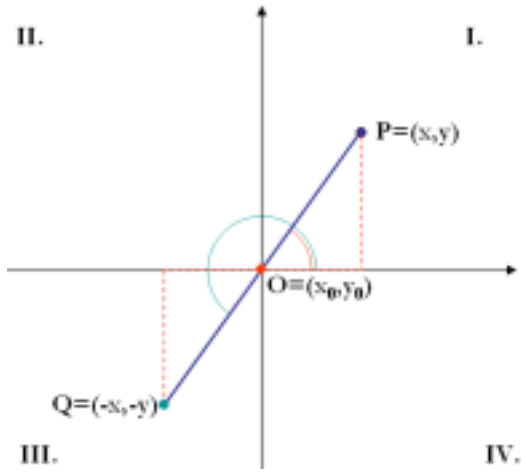
Az oldalt duplavonallal megjelölt kódrész hatékonyabban is megkonstruálható. Az ötlet:

Az iránytengensek kiszámítása és relációba állítása helyett egy (matematikailag) ekvivalens és hatékonyan kiszámítható formulán alapuló relációval helyettesíthető:

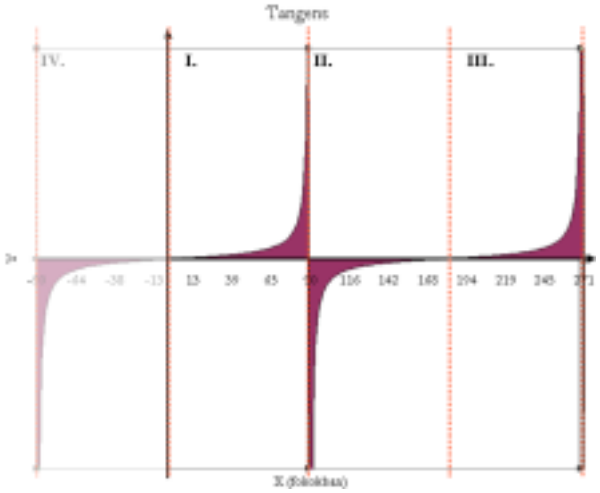
```
      {legbal-alul levőnek az 1. helyre tétele:}
      q:=p[legbal]; p[legbal]:=p[1]; p[1]:=q;
```

```
    ir:=polSzog[legbal]; polSzog[legbal]:=polSzog[1]; polSzog[1]:=ir;
    {a többi polárszög szerinti rendezése:}
    For i:=2 to N-1 do
    Begin
      mini:=i;
      For j:=i+1 to N do
      Begin
        sj:=(p[j].y-p[1].y)*(p[mini].x-p[1].x);
        smini:=(p[mini].y-p[1].y)*(p[j].x-p[1].x);
        If (sj<smini) or
           ((sj=smini) and (p[j].x>p[mini].x)) or
           ((sj=smini) and (p[j].x=p[mini].x) and (p[j].y>p[mini].y))
           {p[j]>p[mini]} then mini:=j
      End;{For j}
      q:=p[i]; p[i]:=p[mini]; p[mini]:=q;
  End;{For i}
```

2. FELADAT – B-MEGOLDÁS



Az iránytangensek nem egyértelműsége.



A tangens függvény síknegyedenkénti monoton növekedése.

A pontok egy másik sorrendjét kaphatjuk meg, ha egy „belső pontból” nézve rendezzük irányszögük szerint. Két kérdést vet föl az ötlet. 1.) mi legyen a belső pont, 2.) az irányszög szerinti rendezést bonyolítja a tg-függvény π/2-kénti szakadásai, és a 3.) π-kénti periodikussága, hiszen az 1. és 3. síknegyedbe „mutató” (π-vel eltérő) egyenes iránytangense azonos, hasonlóan a 2. és 4. síknegyed esetéhez.

Az 1.) megoldására kínálkozik a ponthalmaz súlypontja, ami garantáltan belső pont.

A 2.) megoldása (vagy elkerülése) ugyanaz, mint ami volt az előző megoldás esetében: a π/2 külön kezelése (vagy a tg helyett az ekvivalens reláció használata); szerencsére a π/2 egész számú többszöröseinél van a síknegyedek határa is.

A 3.) megoldásának ötlete, hogy a pontokhoz az iránytangens mellett a síknegyedét is hozzárendeljük: $P \rightarrow (s,t)$, ahol $s \in [1..4]$ – síknegyed, $t \in (-\infty,+\infty)$ – iránytangens; az s meghatározható a P–O koordinátáinak előjeléből ($\mathrm{sgn}(x-x_0)$, $\mathrm{sgn}(y-y_0)$), a t a P–O koordinátáiból $((y-y_0)/(x-x_0))$. Így $P<Q$, ha $P_s<Q_s$, vagy $P_s=Q_s$ és $P_t<Q_t$.

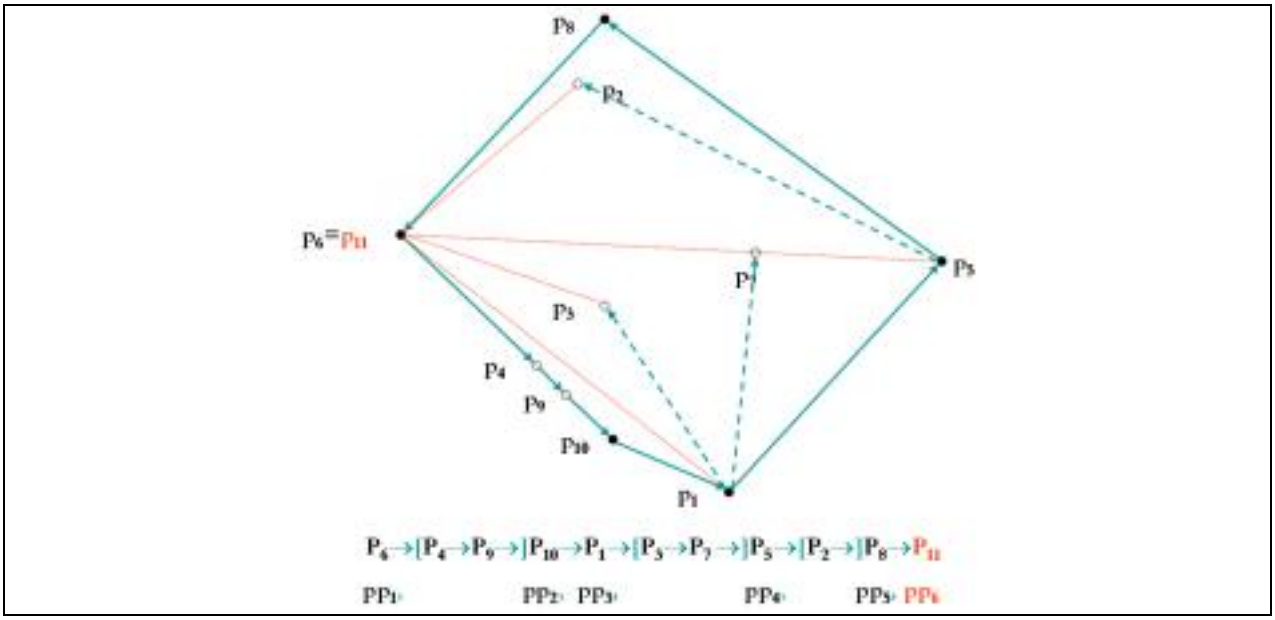
3. FELADAT

Adott N darab (nem kollineáris) pont. Adjuk meg a pontok konvex burkát!

3. FELADAT – MEGOLDÁS

Demó. (KONBUROK.EXE)

Az alábbiakban csak a lényegi eljárás algoritmusát részletezzük:



Az algoritmus által követett pontsorrend (a nyilak jelzik), és ahogy a konvexburokba (pp) bekerülnek az egyes pontok. Szaggatott nyilak utólag nem helyesnek bizonyuló választásokat jelölnek. A p_{11} a többletként előre bekerül végpont.

```
Procedure KonvexBurok(Const N:Integer; Var p:TPontok{polárszög-rendezve};
                      Var   M:Integer; Var pp:TPontok{konvex burok});
(*
  Uf: a feladat meghatározása szerinti sorrendben írja ki a pontokat
*)
  Var
    i,ii:Integer;

  Procedure PolarSzogSzerintRendez(Const N:Integer; Var p:TPontok);
    ... ugyanaz, mint az előbbi feladatot megoldó programnál ...
  End;{PolarSzogSzerintRendez}

Begin{KonvexBurok}
  PolarSzogSzerintRendez(N,p);
  p[N+1]:=p[1];
  {az 1. ponttal egy egyenesre esö pontok átlépése:}
  i:=3;
  While ForgasIrany(p[1],p[i-1], p[i])=0 do Inc(i);
  {ezek közül elegendö az 1. és az "utolsó":}
  pp[1]:=p[1]; pp[2]:=p[i-1];
```

```
    {a többi pont feldolgozása, hármasával:}
    M:=2;
    While (i<=N+1) do
    Begin
      If ForgasIrany(pp[M-1],pp[M],p[i])>=0
         {az M. pontnál nem konvex} then {kihagyható}
      Begin
        Dec(M)
      End
        else
      Begin {vegyük hozzá az újat}
        Inc(M); pp[M]:=p[i];
        Inc(i);
      End;{If ForgasIrany}
    End;
    Dec(M);
    {a burok pontjai:}
    For i:=1 to M do
    Begin
      WritelnTPont(pp[i]);
    End; {For i}
  End;{KonvexBurok}
```

KELLÉKEK – LETÖLTÉSEK

A feladatok megoldásának főprogramjai (a Turbo Pascal néhány specialitását kihasználva):

- Szakaszmetszést vizsgálatának forrása: Metszes.pas;
- Poligont készítő forrása: Pontsorr.pas;
- Konvex burok forrása: Konburok.pas.

A feladatokhoz felhasznált „modulok” (unit-ok, include-állományok):

- A klaviatúra/fájl-inputhoz: Crt22.pas;
- Általános rutinok: Altrutin.inc;
- A grafikus megjelenítés rutinjai: Geograf.inc;
- A pont és a szakasz típusát leírása: Szakpont.inc.

Adatfájlok (*.dat):

- Metszesa.dat, Metszesb.dat, Metszesc.dat;
- Pontok0.dat, Pontok1.dat, Pontok5a.dat, Pontok5b.dat, Pontok5c.dat, Pontok6a.dat, Pontok6b.dat, Pontok6c.dat, Pontok8.dat, Pontok9.dat.

Minden együtt: Geometria.zip.