Programozás
5. előadás

Tartalom

- Típusdefiniálás –
  adatabsztrakció
- Szöveg és tömb –
  összevetés + szöveg feladatok
- Összetett típusok –
  kitekintés
- Függvények –
  algoritmikus absztrakció

A típus fogalma
egy kis összefoglaló

A típus:

- értékhalmaz
- műveletlhalmaz

Rekord-típus:

- **Típus**

  TR=**Rekord(**

  m_1:TM_1,

  …

  m_N:TM_N**)**

- • **:=** •
  • **.**m_1

  …

  • **.**m_N

A típus fogalma

egy kis összefoglaló

Rekord-típus:

- Típus

  TR=**Rekord(**

  m_1:TM_1**,**

  …

  m_N:TM_N**)**

- • **:=** •
  - • **.**m_1

    …
  - • **.**m_N

C++ struktúra-típus:

- typedef

  struct {

  TM_1 m_1**;**

  …

  TM_N m_N**;}** TR**;**

- • **=** •
  - • **.**m_1

    …
  - • **.**m_N

A típus fogalma
egy kis összefoglaló

A típus:

- értékhalmaz
- műveletthalmaz

Tömb-típus:

- **Típus**

  TT=**Tömb[**
  TInd**:**
  TElem**]**

- • **:=** •
  • **[** • **]**

A típus fogalma

egy kis összefoglaló

Tömb-típus:

- Típus
  TT=Tömb[
  TInd:
  TElem]

- • := •
  • [•]

C++ tömb-típus:

- typedef
  TElem TT[
  Számosság”TInd];

- • = •
  • [• – Min”TInd]

A típus fogalma példa

5. Adjuk meg N születésnap alapján azt, akinek idén **először** van születésnapja!

Specifikáció:

- Bemenet: $N \in \mathbb{N}$,
  $Szül_{1..N} \in (hó \times nap)^N$, $hó,nap = \mathbb{N}$
- Kimenet: $Első \in \mathbb{N}$
- Előfeltétel: $N>0$ és
  $\forall i(1 \le i \le N)$: $(Szül_i.hó \in [1..12]$ és
  $Szül_i.nap \in [1..31])$
- Utófeltétel: $1 \le Első \le N$ és
  $\forall i(1 \le i \le N)$: $(Szül_{Első}.hó < Szül_i.hó$ vagy
  $Szül_{Első}.hó = Szül_i.hó$ és
  $Szül_{Első}.nap \le Szül_i.nap)$

Specifikáció:
- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N)$: $X_{Max} \ge X_i$

A típus fogalma
példa

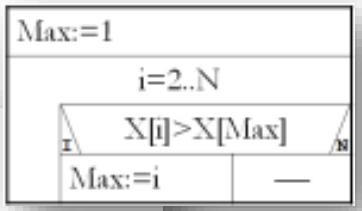
Specifikáció$_2$:

- Bemenet: $N \in \mathbf{N}$,
  $Szül_{1..N} \in Dátum^N$, $Dátum = hó \times nap$, $hó, nap = \mathbf{N}$
- Kimenet: $Első \in \mathbf{N}$
- Előfeltétel: $N > 0$ …
- Utófeltétel: $Első = \min_{i=1}^{N} Ind\ Szül_i$
- Definíció:
  $\leq : Dátum \times Dátum \rightarrow L$
  $d1 \leq d2 \leftrightarrow$ d1.hó<d2.hó vagy
  d1.hó=d2.hó és d1.nap≤d2.nap

5. Adjuk meg N születésnap alapján azt, akinek idén **először** van születésnapja!

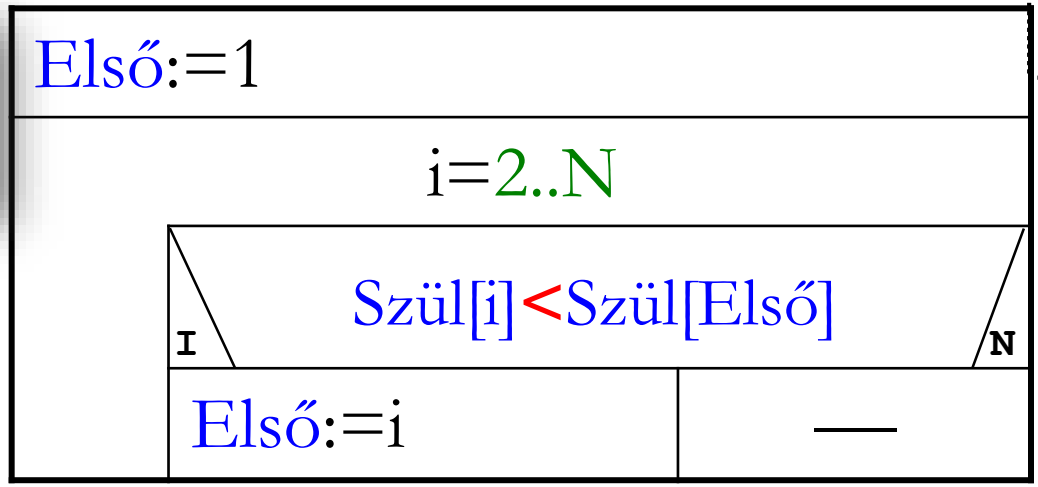
A típus fogalma példa

Algoritmus:



Specifikáció:

- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N)$: $X_{Max} \ge X_i$



Változó
i:Egész

Természetesen meg kell még írni a **Dátum** típushoz a < relációt megvalósító függvényt (operátort).

Az világos, hogy a specifikációban felbukkanó ≤ és az algoritmusbeli < a relációk egymással kifejezhetők.

Szöveg típus

A szöveg és a tömb

- **hasonlóak:**
  - egyféle típusú elemekből állnak,
  - indexelhetők;
- **különbözőek:**
  - a tömb elem- (és index-) típussal paraméterezendő (típuskonstrukciós eszköz), a szöveg karakter típusú elemekből áll,
  - algoritmikusan (!) a szöveg 1-től indexelhető, a tömb deklarációtól függően,
  - a klasszikus tömb hossza konstans, a szövegé változtatható (a dinamikus tömbhöz hasonlóan),
  - szövegeken értelmezve van a hossz() függvény és a + művelet,
  - problémás az elemmódosítás a szöveg típusnál!

Szöveg típus (C++)

- Fontosabb string-műveletek

  (tegyük föl, hogy: `string s;`)

```
""            //üres szöveg; pl.: a=""
cin >> s;     //olvasás szóközig v. sorvégig
getline(cin,s);      //olvasás '\n'-ig
getline(cin,s,'x'); //olvasás 'x' jelig
…s.length()… //az s karakterei száma
…s.size()…   //=s.length()
…+…          //hozzáírás (konkatenáció)
…s[i]…       //s szöveg i. jele, 0≤i<s.length(),
             //túlcímezhető!
…s.at(i)…    //=s[i], objektumos jelöléssel,
             //ellenőrzött!
```

s+t
≡
$s_1+\ldots+s_{hossz(s)}+t_1+\ldots+t_{hossz(t)}$
→
hossz(s+t)=hossz(s)+hossz(t)

Szöveg típus (C++)

- Fontosabb string-műveletek (`string s;`)
  - `…s.find(mit)… //a mit szöveg helye s-ben`
  - `…s.substr(tól,db)… //=s[tól..tól+db-1]`
  - `s.replace(tól,db,mivel); //s-ben tól.-kal`
    `//kezdve db jelet helyettesít a mivel-lel`
- Kapcsolódó char-műveletek (`string s;`)
  - `…isalpha(s[i])… //s[i] 'a'..'z',`
    `//  'A'..'Z'?`
  - `…isdigit(s[i])… //s[i] '0'..'9'?`
  - `…isupper(s[i])… //s[i] 'A'..'Z'?`
  - `…islower(s[i])… //s[i] 'a'..'z'?`
  - `…tolower(s[i])… //s[i]-t kisbetűssé alakítja`
  - `…toupper(s[i])… //s[i]-t nagybetűssé alakítja`

Szöveg feladatok

Feladat:

Fordítsuk meg egy szó (szöveg) betűsorrendjét!

Specifikáció: (másolás tétel)

- Bemenet: $S \in \mathbb{S}$
- Kimenet: $T \in \mathbb{S}$
- Előfeltétel: –
- Utófeltétel: $hossz(T)=hossz(S)$ és
  $\forall i(1 \le i \le hossz(S)): T_i = S_{hossz(S)-i+1}$

Előre definiált függvény:
hossz: $\mathbb{S} \to \mathbb{N}$
hossz(s):=s *karaktereinek a száma*

Szöveg feladatok

Algoritmus: (sorozatszámítás tétel!)

A szöveg i-edik karaktere nem módosítható, ha még nincs.
Szükséges a + művelet!

Specifikáció:

- Bemenet: $S \in S$
- Kimenet: $T \in S$
- Előfeltétel: –
- Utófeltétel: hossz(T)=hossz(S) és $\forall i\ (1 \le i \le \text{hossz}(S)): T_i = S_{\text{hossz}(S)-i+1}$

```
T:=""
i=hossz(S)..1; –1-esével
    T:=T+S[i]
```

Változó
i:Egész

vagy

```
T:=""
i=1..hossz(S)
    T:=S[i]+T
```

Változó
i:Egész

Szöveg feladatok

Feladat:

Adjuk meg egy egyszerű angol névhez a monogramját (pl. James Black ⇒ JB)!

Specifikáció:

- Bemenet: Név∈𝕊
- Kimenet: Mon∈𝕊
- Előfeltétel: SzabályosE(Név)

Megjegyzés: a név szabályos, ha csak a szó kezdőbetűk nagyok, de azok biztosan… feltesszük, hogy definiált a „SzabályosE:𝕊→𝕃” függvény.

Szöveg feladatok

- Bemenet: Név∈S
- Kimenet: Mon∈S
- Előfeltétel: SzabályosE(Név)

- Utófeltétel: $\text{hossz(Mon)} = \sum_{\substack{i=1 \\ \text{NagybetüE}(\text{Név}_i)}}^{\text{hossz(Név)}} 1$ és

  $\forall i\ (1 \le i \le \text{hossz(Mon)})$: NagybetűE($\text{Mon}_i$) és

  Mon⊆Név

Rövidebben: $\text{Mon} = \text{Kiválogat}_{\substack{i=1 \\ \text{NagybetüE}(\text{Név}_i)}}^{\text{hossz(Név)}} \text{Név}_i$

- Megjegyzések:

  1. létezik a „NagybetűE:𝕂→𝕃” függvény,

  2. „⊆” művelet, az ún. részsorozata-e művelet (x ⊆ y akkor igaz, ha x megkapható legfeljebb az y bizonyos elemeinek elhagyásával).

Szöveg feladatok

Algoritmus:

- Utófeltétel: $\text{hossz(Mon)} = \sum_{i=1}^{\text{hossz(Név)}} 1_{\text{NagybetűE(Név}_i\text{)}}$ és

$\forall i\ (1 \le i \le \text{hossz(Mon)})$: NagybetűE(Mon$_i$) és Mon$\subseteq$Név

hossz(Mon)=0

Változó
i:Egész

Mon:=""

i=1..hossz(Név)

I | NagybetűE(Név[i]) | N

Mon:=Mon+Név[i] | –

Mon+Név$_i$≡
≡Mon$_1$+...+Mon$_{\text{hossz(Mon)}}$+Név$_i$
→
hossz(Mon+Név$_i$)=hossz(Mon)+1

Szöveg feladatok

Feladat:

Adjuk meg egy magyar névhez a monogramját (pl. Szabó Éva ⇒ SzÉ)!

Specifikáció:

- Bemenet: Név∈S
- Kimenet: Mon∈S
- Előfeltétel: SzabályosE(Név)
- Utófeltétel:… hf … ☺

Probléma: a monogramban nagybetűk szerepelnek, valamint a kettős mássalhangzókból a nagybetűt követő kisbetűk.

Szöveg feladatok

Adatábrázolás:

```
Konstans  Többes:Tömb[1..8:Szöveg]=
            ("Cs","Dz","Gy","Ly","Ny",
             "Sz","Ty","Zs")
```

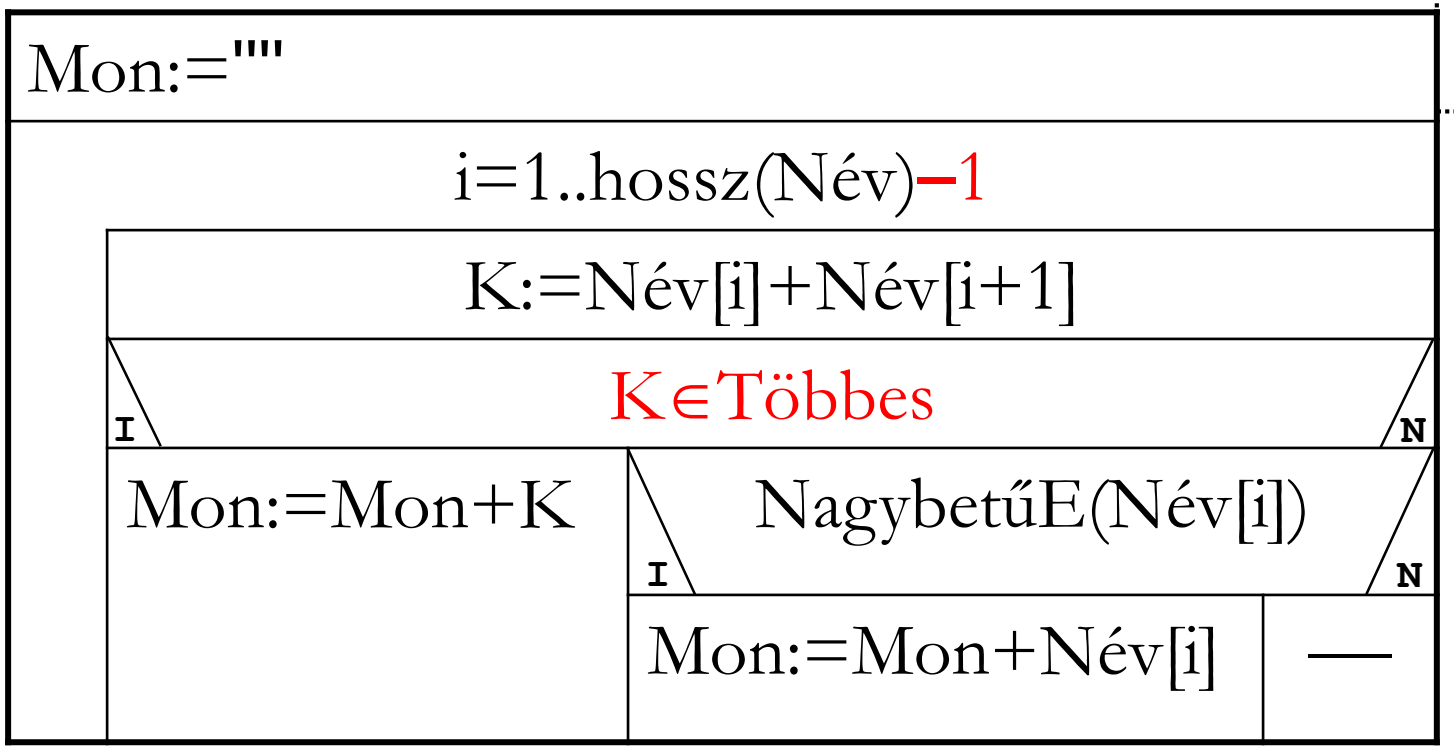
Az algoritmus vázlata:

képezzük a Név kétbetűs részeit, és megnézzük, hogy bent van-e a Többesben, vagy a betűpár első jele nagybetűs-e, ha igen, akkor monogramhoz írjuk…

Szöveg feladatok

Algoritmus:

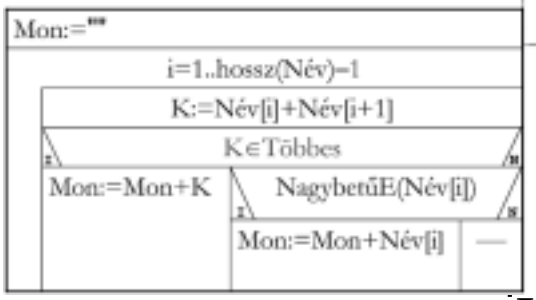
Változó
i:**Egész**
K:**Szöveg**



Problémák:

1) Hívhatnak-e valakit „Nagy A”-nak? 2) Fölcserélhetők-e a feltételek? 3) Optimális-e?

Szöveg feladatok



Algoritmus hatékonyabban:

Változó
i:Egész
K:Szöveg

Mon:=""; i:=1

i≤hossz(Név)−1

K:=Név[i]+Név[i+1]

K∈Többes

I | N

Mon:=Mon+K
i:=i+2

NagybetűE(Név[i])

I | N

Mon:=Mon+Név[i] | —

i:=i+1

Karakter-rendezés

Feladat:

Döntsük el, hogy az A vagy a B betű van-e a magyar ábécében előbb!

Specifikáció:

- Bemenet: $A,B \in \mathbb{K}$
- Kimenet: $Előbb \in \mathbb{L}$
- Előfeltétel: BetűE(A) és BetűE(B)
- Utófeltétel: $Előbb = A <_M B$
- Definíció:

  $x <_M y$ akkor és csak akkor, ha ???
- Feltételezés: $\exists$BetűE:$\mathbb{K} \rightarrow \mathbb{L}$ függvény

Karakter-rendezés

Megoldásötlet:

Tároljuk a helyes sorrendben a betűket, és amelyiket előbb lehet megtalálni, az legyen az előbbi.

→ Kiválasztás tételt alkalmazunk kétszer!

- Definíció:

Betűk∈$\mathbb{K}^{2*35}$=

("a","A","á","Á","b","B",…,"z","Z")

Konstans Betűk:Tömb[1..2*35:**Karakter**]=

("a","A","á","Á","b","B",…,"z","Z")

Egy másik megoldás alapulhat egy szövegen is:

Betűk∈$\mathbb{S}$="aAáÁbB…zZ"

Konstans Betűk:**Szöveg**=("aAáÁbB…zZ")

Karakter-rendezés

Algoritmus:

Specifikáció:
- Bemenet: A,B∈𝕂
- Kimenet: Előbb∈𝕃
- Előfeltétel: BetűE(A) és BetűE(B)
- Utófeltétel: Előbb=A<$_M$B

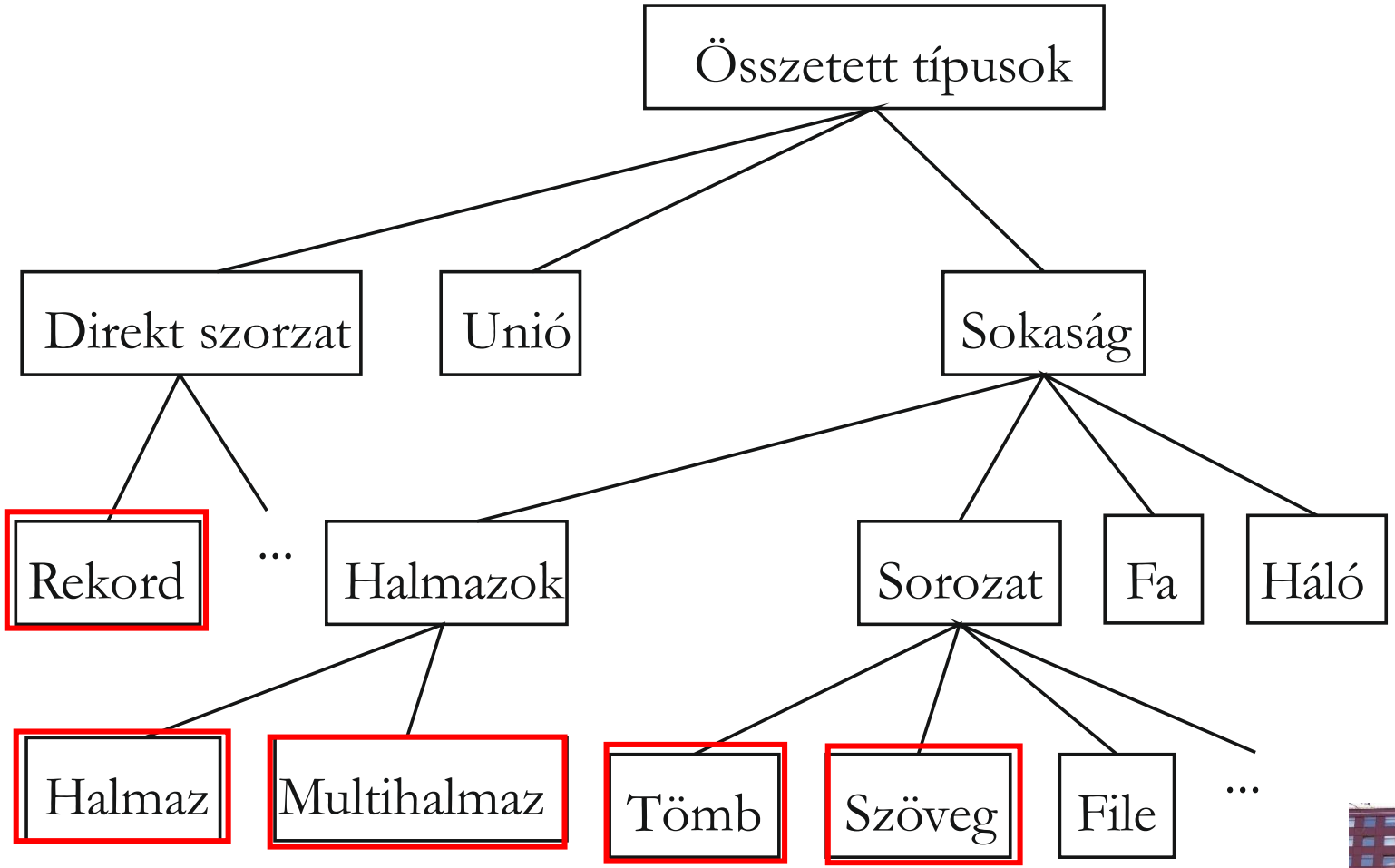
- Definíció:
  Betűk∈$\mathbb{K}^{2*35}$=
  ("a","A","á","Á","b","B",…,"z","Z")
  x<$_M$y akkor és csak akkor, ha
  i<j : x=Betűk$_i$ és y=Betűk$_j$

```
i:=1
Betűk[i]≠A
    i:=i+1
j:=1
Betűk[j]≠B
    j:=j+1
Előbb:=i<j
```

Változó
i,j:**Egész**

Problémák: 1) Mi lenne, ha az előfeltétel nem teljesülne?
2) Lehetne-e "a"="A"? (i-1) div 2<(j-1) div 2

Összetett típusok

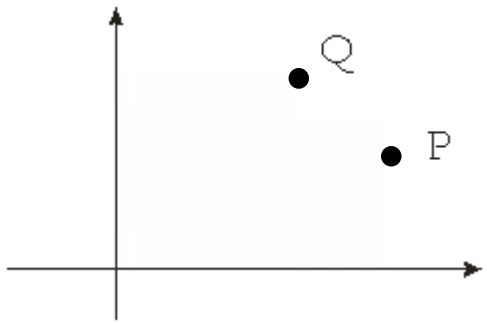


Függvények
(irány)

Feladat:

Adjuk meg, hogy az origóból nézve az 1. síknegyedbe eső P ponthoz képest a Q balra, jobbra, vagy pedig egy irányban látszik-e!

$$\text{Irány}(P,Q) = \begin{cases} -1, & \text{ha balra} \\ +1, & \text{ha jobbra} \\ 0, & \text{ha egy irányban} \end{cases}$$

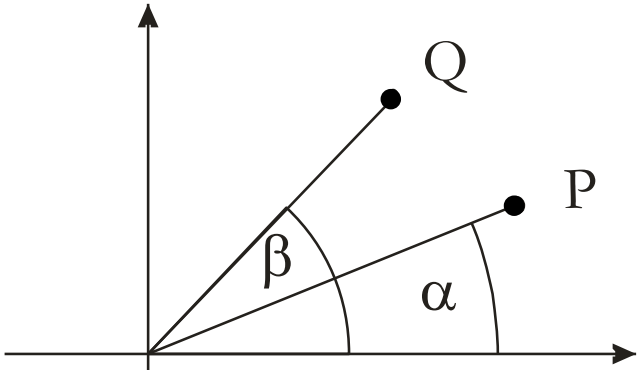


Függvények
(irány)

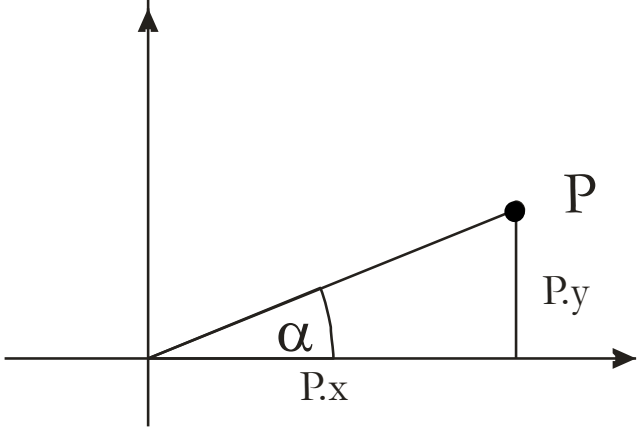
Értelmezés:

A pontok irányát megadhatjuk az origóból oda vezető egyenes és az x-tengely szögével.

$\alpha < \beta \rightarrow \tan(\alpha) < \tan(\beta)$



$\tan(\alpha) = P.y / P.x$



Függvények
(irány)

$\alpha<\beta \leftrightarrow \tan(\alpha)<\tan(\beta) \leftrightarrow$

$P.y/P.x<Q.y/Q.x \rightarrow$

$P.y*Q.x<Q.y*P.x \leftrightarrow$

$P.y*Q.x-Q.y*P.x<0$

Állítás:

Irány(P,Q)=sgn(P.y*Q.x–Q.y*P.x)

(és ez igaz nem csak az 1. síknegyedben).

Ellenőrizze a teljesülését:

$$\text{sgn}(P.y*Q.x-Q.y*P.x) = \begin{cases} -1, & \text{ha Q a P - töl balra} \\ +1, & \text{ha Q a P - töl jobbra} \\ 0, & \text{ha Q és P egy irányban} \end{cases}$$

Függvények
(irány)

Specifikáció:

- Bemenet: $P,Q \in Pont$
  $Pont = x \times y,\ x,y = \mathbb{Z}$
- Kimenet: $Ir \in \mathbb{Z}$
- Előfeltétel: –
- Utófeltétel: $Ir = Irány(P,Q)$
- Definíció: $Irány: Pont \times Pont \rightarrow \mathbb{Z}$
  $Irány(p,q) := sgn(p.y*q.x - q.y*p.x)$

Függvény-szignatúra:
fv : *Honnan*→*Hova*

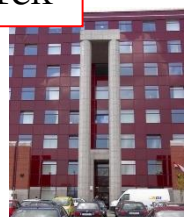
Aktuális paraméterek

Formális paraméterek

Algoritmus:

```
Ir:=Irány(P,Q)
```

Aktuális paraméterek



Függvények
(irány)

Az **Irány** függvény kiszámítását tekintsük **önálló feladat**nak!

Definíció: Irány:Pont×Pont→Z
Irány(p,q):=sgn(p.y*q.x–q.y*p.x)

Specifikáció:

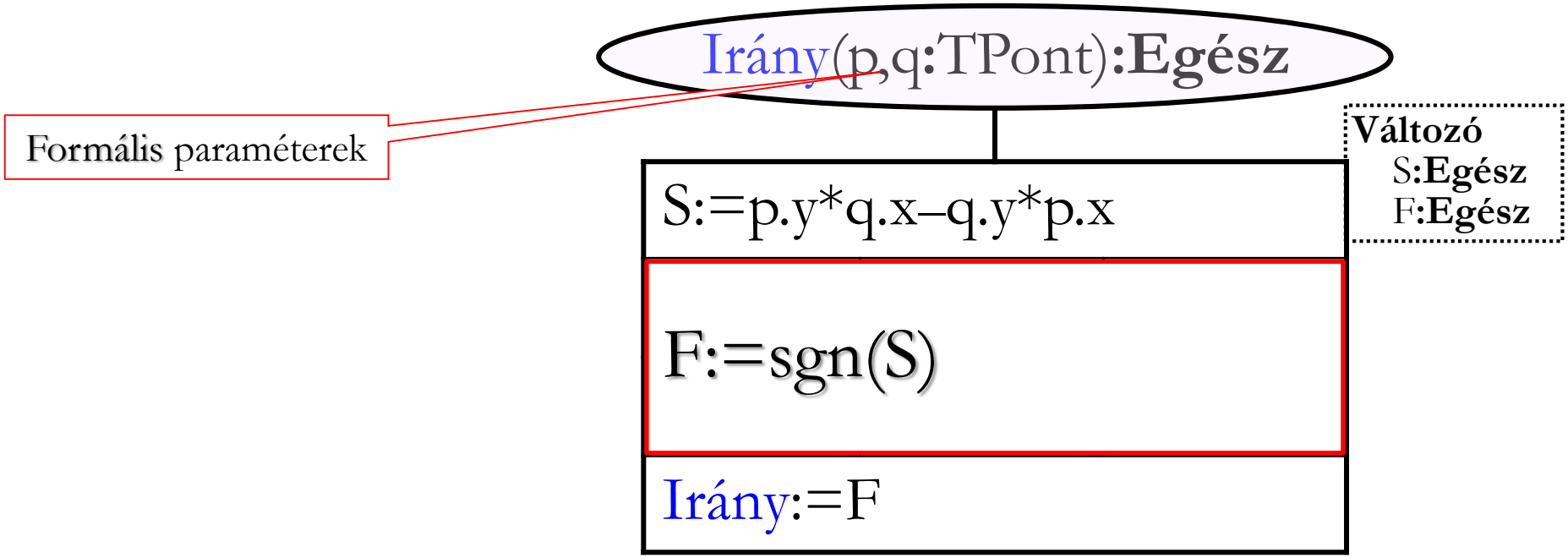
- Bemenet: p,q∈Pont, Pont=x×y, x,y=ℤ
- Kimenet: Irány(p,q)∈ℤ
- Előfeltétel: –
- Utófeltétel: Irány(p,q)=sgn(p.y*q.x–q.y*p.x)

A bemenetben a függvény paraméterei (∈értelmezési tartomány), a kimenetben a függvény paraméteres értéke szerepel (∈értékkészlet), az utófeltételben az összefüggés.

Függvények
(irány)

- **Specifikáció → algoritmus**
  - **függvénydefiníció:**

Definíció: Irány:Pont×Pont→Z
Irány(p,q):=sgn(p.y*q.x–q.y*p.x)



Fogalmak
(C++)

Blokk. A { és a hozzá tartozó } közötti programszöveg.

Hatáskör. Egy X azonosító hatásköre az a programszöveg (nem feltétlenül összefüggő), ahol az azonosítóra hivatkozni lehet.

A hatáskört a blokkstruktúra határozza meg. Ha két blokknak van közös része, akkor az egyik teljes egészében tartalmazza a másikat.

Egy azonosító hatásköre a deklarációját követő karaktertől a blokkot lezáró } végzárójelig tart, kivéve azt a beágyazott blokkot, és ennek beágyazottjait, amely-ben újra lett deklarálva.

```
int main()
{
  int i=1,j=2;
  for (int i=1;…){
    cout<<i…;
    j++;
    …
  }
  cout<<i+j…;
  …
}
```

Fogalmak
(C++)

Hatáskör. Egy adott helyen hivatkozott azonosító lokális, ha a hivatkozás helyét tartalmazó legszűkebb blokkban lett deklarálva. Egy azonosító globális (az adott blokkra nézve), ha nem lokális.

Élettartam. Minden B blokkban deklarált változó élettartalma a blokkba való belépéstől a blokk utolsó utasításának befejeződéséig tart.

```
int main()
{
   int i=1,j=2;
   for (int i=1;…){
      cout<<i…;
      j++;
      …
   }
   cout<<i+j…;
   …
}
```

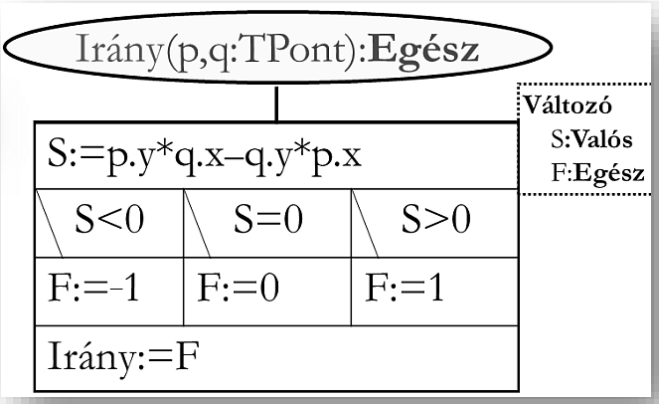
Függvények
(irány)

- **Algoritmus → kód:**
  - A főprogramban függvényhívás:

Ir:=Irány(P,Q)

```
Ir=Irany(P,Q);
```

  - A függvény definiálása:




```
int Irany(TPont p, TPont q)
{
 int S; int F; //segédváltozók
 S=p.y*q.x-q.y*p.x;
 if (S<0) F=-1;
 else if (S==0) F=0;
 else if (S>0) F=1;
 return F;
}
```

Függvények

C++ tudnivalók – összefoglalás:

- Függvényfej-definíció (prototípus):

```
fvTíp fvAzon(parTíp formParAzon,…);
        vagy
void fvAzon(parTíp formParAzon,…);
```

A formális paraméterek elmaradhatnak, de a zárójelek nem!

- Függvény-definíció:

A fejsor, a végén pontosvessző nélkül

```
{
   … //fvtörzs
   return fvÉrték;
}
```

`void` esetén `fvÉrték` nélküli `return`, vagy `return` nélkül!

Függvények

C++ tudnivalók – összefoglalás:

- **Formális** skalár paraméter:
  - bemeneti → nincs speciális kulcsszó
  - kimeneti → & a speciális prefix „kulcsszó”

Pontosabban: **nem tömb**

- **Aktuális** skalár paraméter:

ha a megfelelő formális paraméter

  - bemeneti → akár konstans, akár változó
  - kimeneti → **csak** változó

lehet.

Pontosabban: **nem tömb**

Függvények

C++ tudnivalók – összefoglalás:

- Skalár paraméterátadás:
  - **Értékszerinti** – a formális paraméterből „keletkezett” lokális változóba másolódik a híváskor az aktuális paraméter értéke, így ennek a törzsön belüli megváltozása nincs hatással az aktuális paraméterre. Pl.:

Input-paraméterek.

```
int max(int x, int y)
```

  - **Hivatkozás szerinti** – a formális paraméterbe az aktuális paraméter címe (*rá való hivatkozás*) kerül, a lokális néven *is* elérhetővé válik. Pl.:

```
void max(int x, int y, int& max_xy)
```

Input-paraméterek.

In-/Output-paraméter.

Függvények

C++ tudnivalók – összefoglalás:

- Tömb paraméterátadás:
  Alapelv: a tömbök mindig hivatkozás szerint adódnak át!

- **Formális** tömb paraméter:
  - bemeneti → **const** prefix kulcsszó
  - kimeneti → nincs speciális kulcsszó
- **Aktuális** tömb paraméter:
  ha a megfelelő formális paraméter
  - bemeneti → akár konstans, akár változó
  - kimeneti → **csak** változó

  lehet.

Finomítások a C++ kódban

C++ tudnivalók – összefoglalás:

- Bemeneti paraméter – példa a fejsorra:

Input-paraméterek.

```
void ki_int_tomb(const int x[], int n)
```

vagy

```
void ki_int_tomb(const int x[maxN], int n)
```

- Kimeneti paraméter – példa a fejsorra:

Ellenőrzési céllal, input-paraméter

```
void be_int_tomb(const int x[], int& n, int maxN)
```

vagy

```
void be_int_tomb(const int x[maxN],int& n, int maxN)
```

In-/ Output-paraméterek.

Finomítások a C++ kódban

C++ tudnivalók – összefoglalás:

- Bemeneti paraméter – példa a fejsorra:

```
void ki_int_tomb(const int x[], int n)
                        vagy
void ki_int_tomb(const int x[maxN], int n)
```

- Kimeneti paraméter – példa a fejsorra:

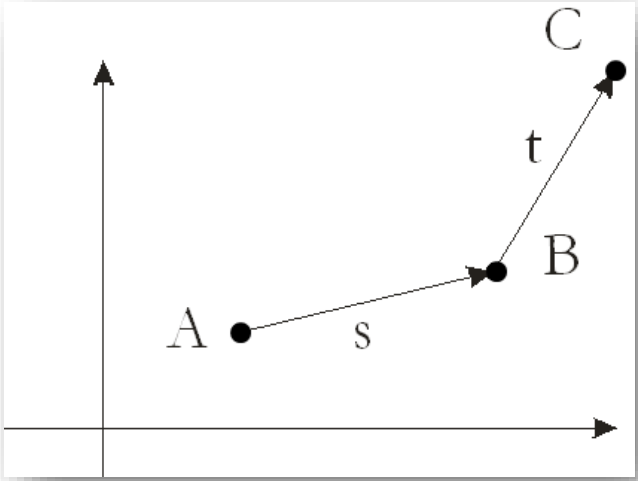
```
void be_int_tomb(int x[], int &n, int maxN)
                        vagy
void be_int_tomb(int x[maxN],int &n, int maxN)
```

Megjegyzés: az 1. változat mátrixokra nem működik!

Függvények
(fordul)

Feladat:

Egy **s** (A→B) szakaszhoz képest egy **t** (B→C) szakasz milyen irányban fordul?

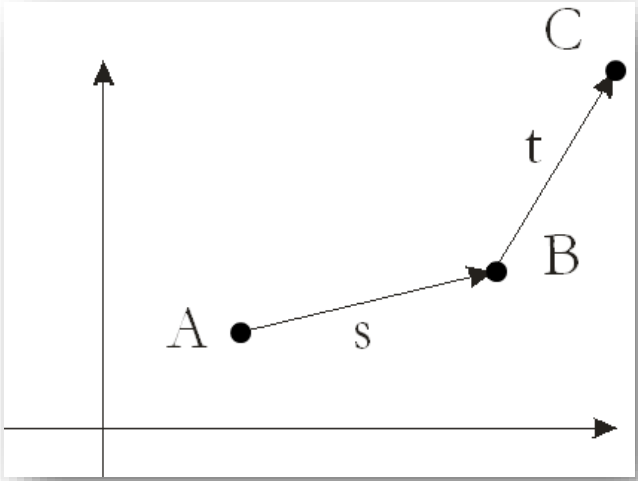


Megoldásötlet:

Toljuk el az **s**-t és a **t**-t úgy, hogy az A pont az origóba kerüljön! Ezzel visszavezetjük az „irányos” feladatra!

Fordul(A,B,C)=Irány(B–A,C–A)

Függvények
(fordul)



Specifikáció:

- Bemenet: A,B,C∈Pont, Pont=…
- Kimenet: Ford∈ℤ
- Előfeltétel: –
- Utófeltétel: Ford=Fordul(A,B,C)
- Definíció: Fordul:Pont^3→ℤ
  Fordul(a,b,c):=Irány(b–a,c–a)
- Megjegyzés:
  Ezzel ekvivalens feladat: *Az (A,B)-n átmenő egyenestől a C pont balra van, vagy jobbra van, vagy az (A,B)-re illeszkedő egyenesen van?*

Függvények
(fordul)

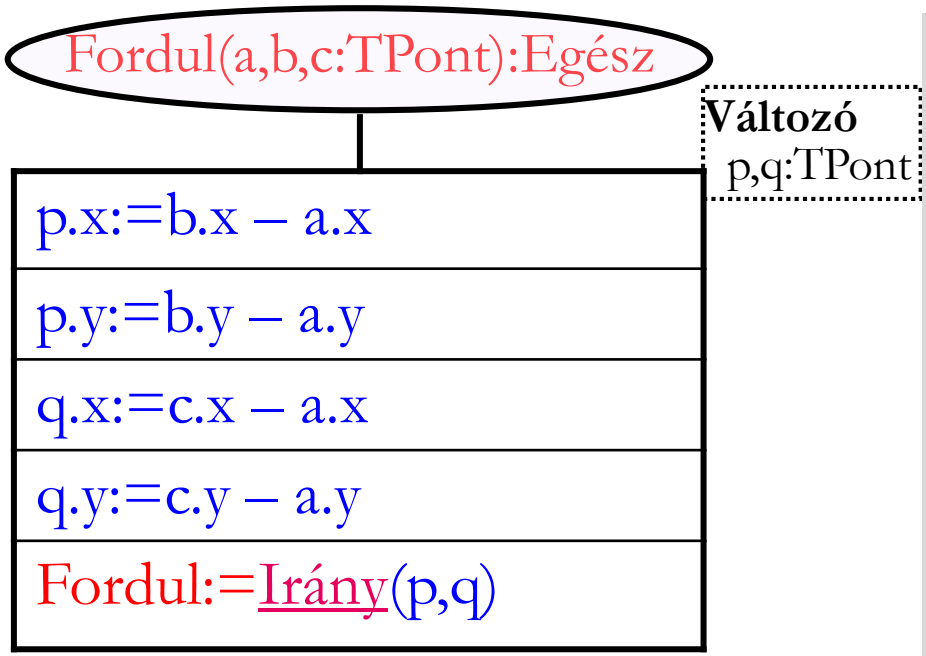
Specifikáció:
- Bemenet: A,B,C∈Pont, Pont=...
- Kimenet: Ford∈Z
- Előfeltétel: –
- Utófeltétel: Ford=Fordul(A,B,C)
- Definíció: Fordul:Pont³→Z
  Fordul(a,b,c):=Irány(b–a,c–a)

Algoritmus:

Ford:=Fordul(A,B,C)

- A megoldásban hívjuk az Irány függvényt!

Finomítás:



Változó
p,q:TPont

```
int Fordul(TPont a, TPont b,
           TPont c){
   TPont p,q;

   p.x=b.x - a.x;

   p.y=b.y - a.y;

   q.x=c.x - a.x;

   q.y=c.y - a.y;

  return Irany(p,q);
}
```

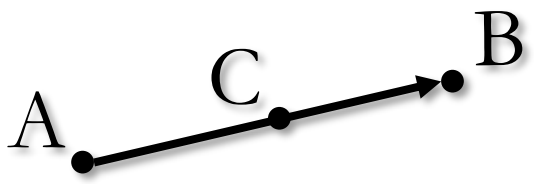
Függvények
(rajta?)

Feladat:

Döntsük el, hogy egy C pont rajta van-e egy (A,B) szakaszon!

Specifikáció:

- Bemenet: $A,B,C \in Pont$
- Kimenet: $RajtaE \in \mathbb{L}$
- Előfeltétel: –
- Utófeltétel: $RajtaE = Rajta(A,B,C)$
- Definíció: $Rajta: Pont^3 \rightarrow \mathbb{L}$
  $Rajta(a,b,c) := \ldots$



Függvények
(rajta?)

Definíció (ami egyben a két függvény specifikációja – utófeltétele)**:**

Rajta:Pont^3→𝕃

Rajta(a,b,c):= Fordul(a,c,b)=0 és
Közte(a.x,c.x,b.x) és
Közte(a.y,c.y,b.y)

Azaz még egy függvényt kell definiálnunk, ami eldönti, hogy a második paramétere a másik kettő között van-e!

Közte:ℤ^3→𝕃

Közte(r,s,t):= r ≤ s ≤ t vagy t ≤ s ≤ r

Függvények
(rajta?)

Specifikáció:
- Bemenet: A,B,C∈Pont
- Kimenet: RajtaE∈L
- Előfeltétel: –
- Utófeltétel: RajtaE=Rajta(A,B,C)
- Definíció: Rajta:Pont³→L
  Rajta(a,b,c):=…

Algoritmus:

RajtaE:=Rajta(A,B,C)

- **Finomítások:**

Rajta(a,b,c:TPont):**Logikai**

Rajta:= Fordul(a,c,b)=0 és
Közte(a.x,c.x,b.x) és
Közte(a.y,c.y,b.y)

Közte(r,s,t:**Egész**):**Logikai**

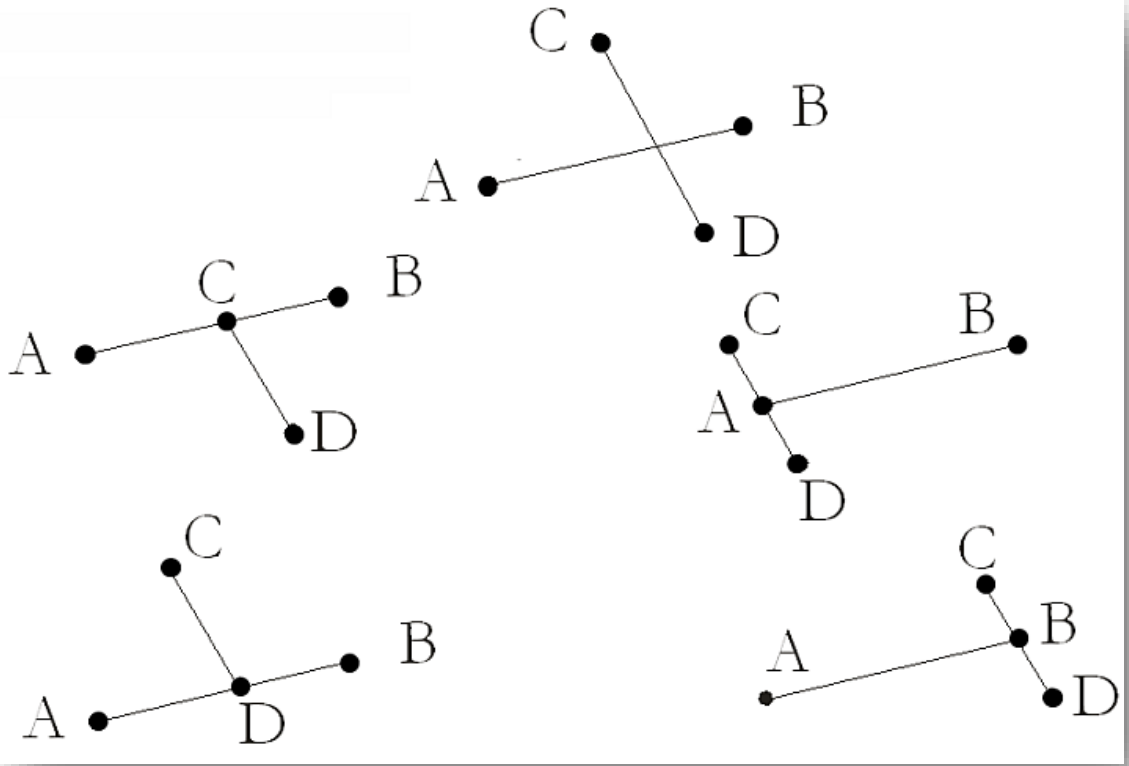
Közte:=r≤s és s≤t vagy t≤s és s≤r

Függvények
(metszi?)

Feladat:

Döntsük el, hogy az (A,B) szakasz metszi-e a (C,D) szakaszt!

Lehetséges esetek:

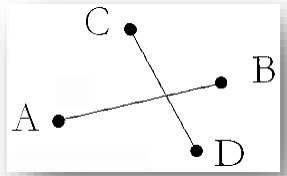


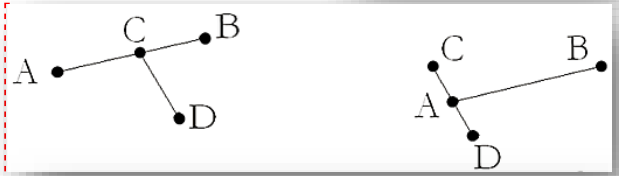
Függvények
(metszi?)

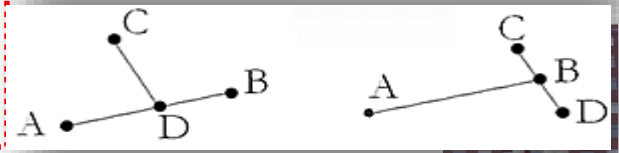
Specifikáció:

- Bemenet: $A,B,C,D \in Pont$
- Kimenet: $MetsziE \in \mathbb{L}$
- Előfeltétel: $A \neq B$ és $C \neq D$
- Utófeltétel: $MetsziE =$ Metszi(A,B,C,D)

(Fordul(A,B,C)*Fordul(A,B,D)<0 és
Fordul(C,D,A)*Fordul(C,D,B)<0
vagy
Rajta(A,B,C) vagy Rajta(C,D,A)
vagy
Rajta(A,B,D) vagy Rajta(C,D,B))







Függvények
(metszi?)

Algoritmus-finomítás:

- Utófeltétel: MetsziE=
  (Fordul(A,B,C)*Fordul(A,B,D)<0 és
  Fordul(C,D,A)*Fordul(C,D,B)<0
  vagy
  Rajta(A,B,C) vagy Rajta(A,B,D)
  vagy
  Rajta(C,D,A) vagy Rajta(C,D,B))

Metszi(a,b,c,d:TPont):**Logikai**

Metszi:=Fordul(a,b,c)*Fordul(a,b,d)<0 és
Fordul(c,d,a)*Fordul(c,d,b)<0
vagy
Rajta(a,b,c) vagy Rajta(a,b,d)
vagy
Rajta(c,d,a) vagy Rajta(c,d,b)

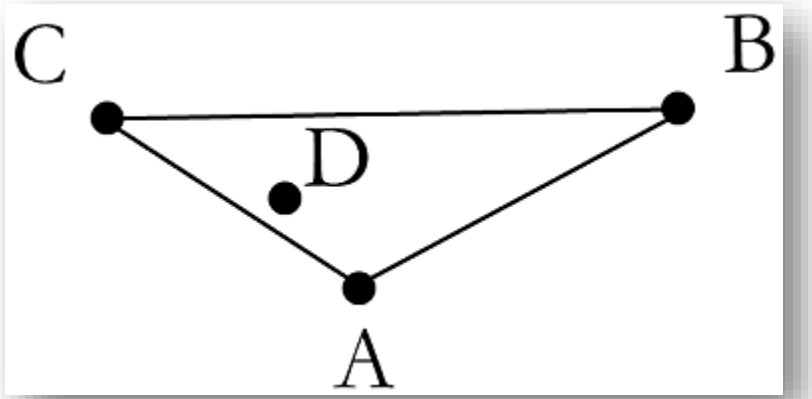
Függvények
(háromszögben?)

Feladat:

Döntsük el, hogy a D pont az (A,B,C) háromszög belsejében van-e!

Megoldásötlet:

Belül van, ha a háromszöget A→B→C→A sorrendben körbejárva a D pont vagy mindig balra, vagy mindig jobbra van.



Függvények
(háromszögben?)

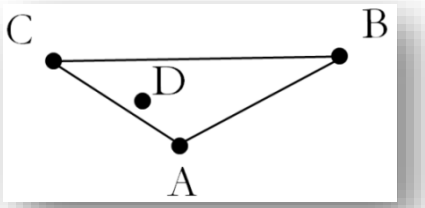
Specifikáció:

- Bemenet: $A,B,C,D \in Pont$
- Kimenet: $BentE \in \mathbb{L}$
- Előfeltétel: $A \neq B$ és $B \neq C$ és $C \neq A$
- Utófeltétel: $BentE = Belül(A,B,C,D)$
- Definíció:

$Belül: Pont^4 \rightarrow \mathbb{L}$

$Belül(a,b,c,d) := Fordul(a,b,d) = Fordul(b,c,d)$

és $Fordul(b,c,d) = Fordul(c,a,d)$

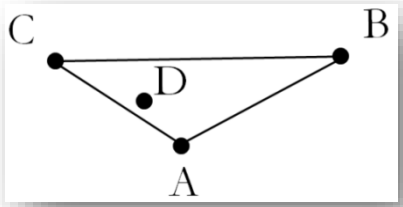


Függvények
(háromszögben?)

Algoritmus-finomítás:

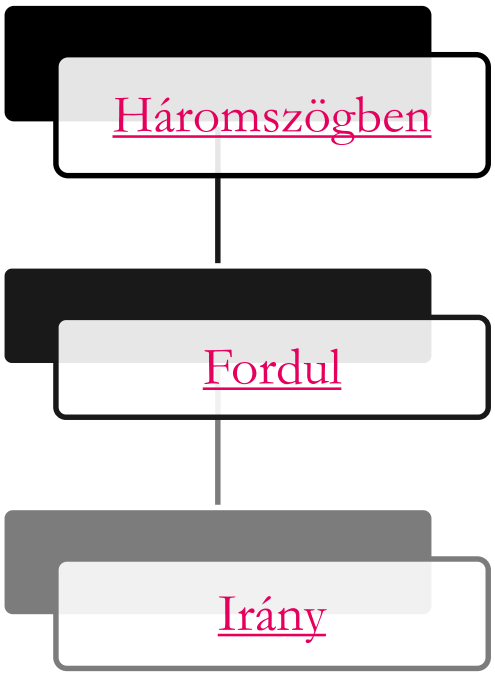
Belül(a,b,c,d:TPont):**Logikai**

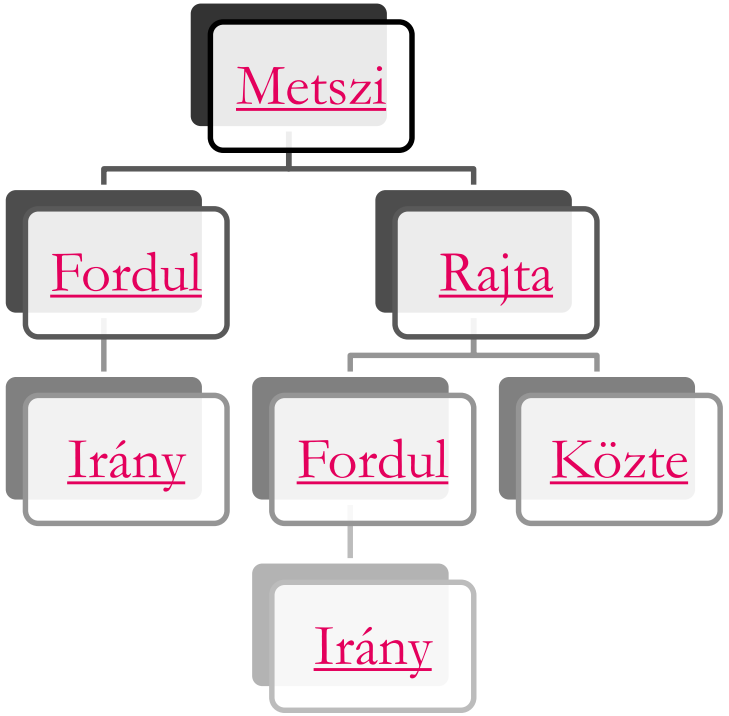
Belül:=Fordul(a,b,d)=Fordul(b,c,d)
és Fordul(b,c,d)=Fordul(c,a,d)



Függvények

A (lényegi) függvények egymásra épülése – a programok makrószerkezete:





Tartalom

- Típusdefiniálás –
  adatabsztrakció
- Szöveg és tömb –
  összevetés + szöveg feladatok
- Összetett típusok –
  kitekintés
- Függvények –
  algoritmikus absztrakció