

An aerial photograph of Budapest, Hungary, showing the Danube River and the city skyline. The image is in black and white, with a semi-transparent white rectangle overlaid in the center containing the text "Programozás 5. előadás".

Programozás

5. előadás

Tartalom



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



A **típus** fogalma

egy kis összefoglaló

A típus:

- értékthalmaz

- művelethalmaz

Rekord-típus:

- Típus

TR=**Rekord**(
 $m_1:TM_1$,
 ...
 $m_N:TM_N$)

- • **:=** •
- $.m_1$
- ...
- $.m_N$

Példa:

TPont=**Rekord**(
 $x, y:Egész$)



A típus fogalma

egy kis összefoglaló



Rekord-típus:

➤ Típus

TR=**Rekord**(
 $m_1:TM_1$,
 ...
 $m_N:TM_N$)

- • **:=** •
- $.m_1$
- ...
- $.m_N$

Példa:

TPont=**Rekord**(
 $x,y:Egész$)

C++ struktúra-típus:

➤ typedef

struct {
 $TM_1 m_1$;
 ...
 $TM_N m_N$;} TR;

- • **=** •
- $.m_1$
- ...
- $.m_N$

Példa:

typedef struct{
 $int x,y$;} TPont;



A típus fogalma

egy kis összefoglaló

A típus:

- értékthalmaz

- művelethalmaz

Tömb-típus:

- Típus

$TT = \text{Tömb}[$
TInd:
TElem]

- • := •
• [•]

Példa:

TÉvszakok = **Tömb** [1 .. 4 : Szöveg]



A típus fogalma

egy kis összefoglaló



Tömb-típus:

➤ Típus

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

- • **:=** •
- **[•]**

Példa:

```
TÉvszakok=Tömb[1..4:Szöveg]
```

C++ tömb-típus:

➤ typedef

```
TElem TT[  
    Számosság'TInd];
```

- • **=** •
- **[• – Min'TInd]**

Példa:

```
typedef string TEvszakok[4];
```



A típus fogalma (példa)



Specifikáció:

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

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

Specifikáció:

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



A típus fogalma (példa)



Specifikáció₂:

- Bemenet: $N \in \mathbf{N}$,
 $Szül_{1..N} \in \text{Dátum}^N$, $\text{Dátum} = \text{hó} \times \text{nap}$, $\text{hó}, \text{nap} = \mathbf{N}$
- Kimenet: $\text{Első} \in \mathbf{N}$
- Előfeltétel: $N > 0 \dots$
- Utófeltétel: $\text{Első} = \underset{i=1}{\overset{N}{\text{MinInd}}} Szül_i$
- Definíció: $\leq : \text{Dátum} \times \text{Dátum} \rightarrow \mathbf{L}$
 $d1 \leq d2 \iff d1.\text{hó} < d2.\text{hó} \text{ vagy } d1.\text{hó} = d2.\text{hó} \text{ és } d1.\text{nap} \leq d2.\text{nap}$



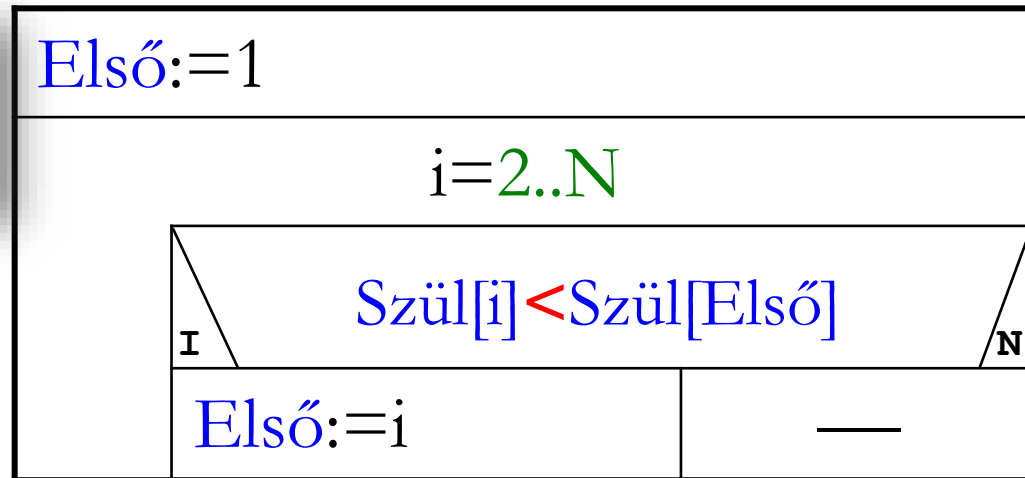
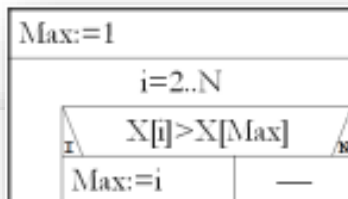
A típus fogalma (példa)

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

Algoritmus:

Specifikáció:

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



Változó
i:Egész

Megjegyzések:

1. Természetesen meg kell még írni a **Dátum** típushoz a $<$ relációt megvalósító függvényt (operátort). L. később!
2. Az is világos, hogy a specifikációban felbukkanó \leq és az algoritmusbeli $<$ a relációk egymással kifejezhetők.

Kóddarab
jegyzet-
ként

A szöveg és a tömb

➤ hasonlóak:

- egyféle típusú elemekből állnak,
- indexelhetők;

➤ különbözőek:

- a tömb elem- (és index-) **típussal paraméterezendő** ún. **típuskonstrukciós eszköz**, a szöveg **karakter típusú** elemekből álló **típus**,
- 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 **szóköz**ig v. **sorvég**ig
- `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!**

$$\begin{aligned} s+t &\equiv s_1+...+s_{\text{hossz}(s)}+t_1+...+t_{\text{hossz}(t)} \\ &\rightarrow \text{hossz}(s+t) = \text{hossz}(s) + \text{hossz}(t) \end{aligned}$$



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



Feladat:

Fordítsuk meg egy szó (szöveg) betűsorrendjét!

Specifikáció: (másolás tétel)

- Bemenet: $S \in \mathbb{S}$
- Kimenet: $T \in \mathbb{S}$
- Előfeltétel: –
- Utófeltétel: $\text{hossz}(T) = \text{hossz}(S)$ és
 $\forall i (1 \leq i \leq \text{hossz}(S)): T_{\text{hossz}(S)-i+1} = S_i$

Előre definiált függvény:

$\text{hossz}: \mathbb{S} \rightarrow \mathbb{N}$

$\text{hossz}(s) := s$ karaktereinek a száma



Szöveg feladatok



Algoritmus: (**sorozatszámítás** tétel!)

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

Specifikáció: (másolás tétel)

- > Bemenet: $S \in \mathcal{S}$
- > Kimenet: $T \in \mathcal{S}$
- > Előfeltétel: –
- > Utófeltétel: $\text{hossz}(T) = \text{hossz}(S)$ és
 $\forall i (1 \leq i \leq \text{hossz}(S)) : T_{\text{hossz}(S)-i+1} = S_i$

$T := ""$

$i = \text{hossz}(S) .. 1; -1\text{-esével}$

$T := T + S[i]$

Változó

i : Egész

vagy

$T := ""$

$i = 1 .. \text{hossz}(S)$

$T := S[i] + T$

Változó

i : Egész



Feladat:

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

Specifikáció:

- Bemenet: $Név \in S$
- Kimenet: $Mon \in S$
- Előfeltétel: SzabályosE(Név)

Megjegyzések:

1. A név szabályos, ha csak a szó kezdőbetűk nagyok, de azok biztosan.
2. Feltesszük, hogy definiált a „SzabályosE: $S \rightarrow L$ ” függvény. L. később!



Szöveg feladatok

- Bemenet: $Név \in S$
- Kimenet: $Mon \in S$
- Előfeltétel: $SzabályosE(Név)$

➤ Utófeltétel:
$$hossz(Mon) = \sum_{\substack{i=1 \\ NagybetűE(Név_i)}}^{hossz(Név)} 1 \quad \text{és}$$

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

Rövidebben:
$$Mon = \bigcup_{\substack{i=1 \\ NagybetűE(Név_i)}}^{hossz(Név)} Név_i$$

Megjegyzések, emlékeztető:

1. A $Név_i$ a $Név$ **szöveg i. jelét** jelöli; a $hossz(Név)$ a $Név$ **szöveg hosszát**.
2. Feltesszük, hogy létezik a „**NagybetűE**: $K \rightarrow L$ ” függvény. L később!
3. A „ **\subseteq** ” művelet, az ún. **részsorozata-e** művelet ($x \subseteq y$ akkor igaz, ha x megkapható legfeljebb az y bizonyos elemeinek elhagyásával).

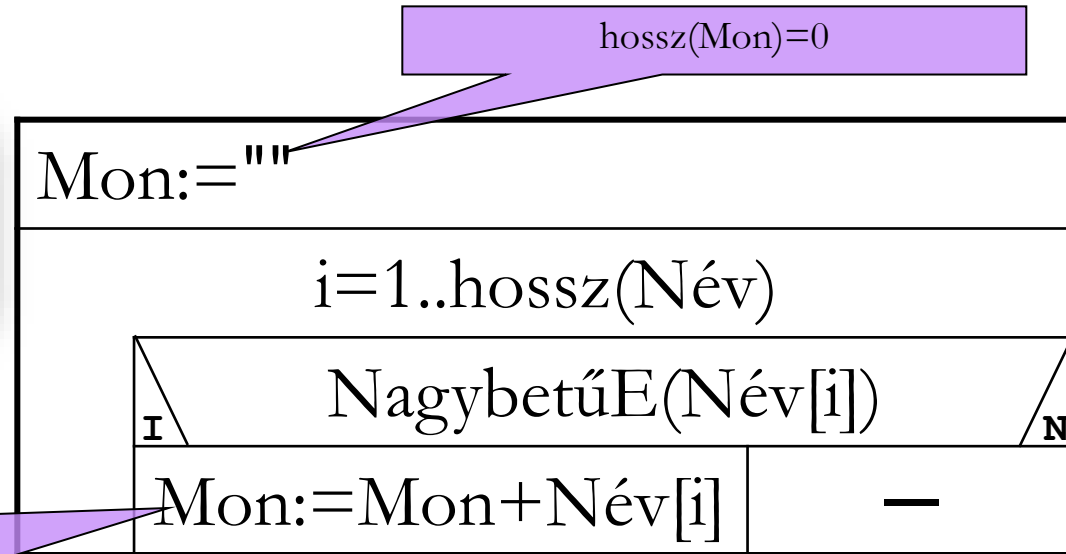


Szöveg feladatok



Algoritmus:

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



Változó
i:Egész

$\text{Mon} + \text{Név}_i \equiv$
 $\equiv \text{Mon}_1 + \dots + \text{Mon}_{\text{hossz}(\text{Mon})} + \text{Név}_i$
 \rightarrow
 $\text{hossz}(\text{Mon} + \text{Név}_i) = \text{hossz}(\text{Mon}) + 1$



Feladat:

Adjuk meg egy magyar névhez a monogramját (pl. **Sz**abó Éva \Rightarrow **SzÉ**)!

Specifikáció:

- Bemenet: $Név \in S$
- Kimenet: $Mon \in S$
- Előfeltétel: SzabályosE(Név)
- Utófeltétel: ... hf ... ☺

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



Adatábrázolás:

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

Az algoritmus vázлата:

képezzük a Név kétbetűs részeit, és megnézzük, hogy

1. bent van-e a Többesben, vagy
2. a betűpár első jele nagybetűs-e,

ha igen, akkor monogramhoz írjuk...



Szöveg feladatok



Algoritmus:

Változó
i:Egész
K:Szöveg

Mon:=""		
i=1..hossz(Név)-1		
K:=Név[i]+Név[i+1]		
I	K ∈ Többses	
Mon:=Mon+K	I	NagybetűE(Név[i])
	Mon:=Mon+Név[i]	—

Problémák:

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



Szöveg feladatok



Változó
i:Egész
K:Szöveg

Mon:=""		
i:=1..hossz(Név)-1		
K:=Név[i]+Név[i+1]		
K∈Többses		
Mon:=Mon+K	NagybetűE(Név[i])	
Mon:=Mon+Név[i]		—

Változó
i:Egész
K:Szöveg

Mon:=""; i:=1

$i \leq \text{hossz}(\text{Név}) - 1$

K:=Név[i]+Név[i+1]

K∈Többses	
-----------	--

Mon:=Mon+K

NagybetűE(Név[i])

i:=i+2

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: $\text{Előbb} \in \mathbb{L}$
- Előfeltétel: $\text{BetűE}(A)$ és $\text{BetűE}(B)$
- Utófeltétel: $\text{Előbb} = A <_{\mathbf{M}} B$
- Definíció:
 $x <_{\mathbf{M}} y$ akkor és csak akkor, ha ???

Megjegyzés:

Feltesszük, hogy létezik a $\text{BetűE}: \mathbb{K} \rightarrow \mathbb{L}$ függvény.



Karakter- rendezés



Megoldásötlet:

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

→ Kiválasztás tételt alkalmazunk kétszer!

➤ Definíció:

$\text{Betűk}_{1..2*35} \in \mathbb{K}^{2*35} = ("a", "A", "á", "Á", "b", "B", \dots, "z", "Z")$

$x <_{\mathbf{M}} y \leftrightarrow i < j : x = \text{Betűk}_i \text{ és } y = \text{Betűk}_j$



Karakter- rendezés



Megoldásötlet (ábrázolás):

a)

$\text{Betűk}_{1..2*35} \in \mathbb{K}^{2*35} = ("a", "A", "á", "Á", "b", "B", \dots, "z", "Z")$



Konstans $\text{Betűk: Tömb}[1..2*35: \mathbf{Karakter}] =$
("a", "A", "á", "Á", "b", "B", ..., "z", "Z")

b)

$\text{Betűk} \in \mathbf{S} = "aAáÁbB...zZ"$



Konstans $\text{Betűk: Szöveg} = ("aAáÁbB...zZ")$



Karakter- rendezés

Algoritmus:

Specifikáció:

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

Definíció:

$Betűk \in K^{2*35} =$

$(\text{"a"}, \text{"A"}, \text{"á"}, \text{"Á"}, \text{"b"}, \text{"B"}, \dots, \text{"z"}, \text{"Z"})$

$x <_M y$ akkor és csak akkor, ha

$i < j : x = Betű_k_i$ és $y = Betű_k_j$

Változó

i, j : Egész

$i := 1$

$Betűk[i] \neq A$

$i := i + 1$

$j := 1$

$Betűk[j] \neq B$

$j := j + 1$

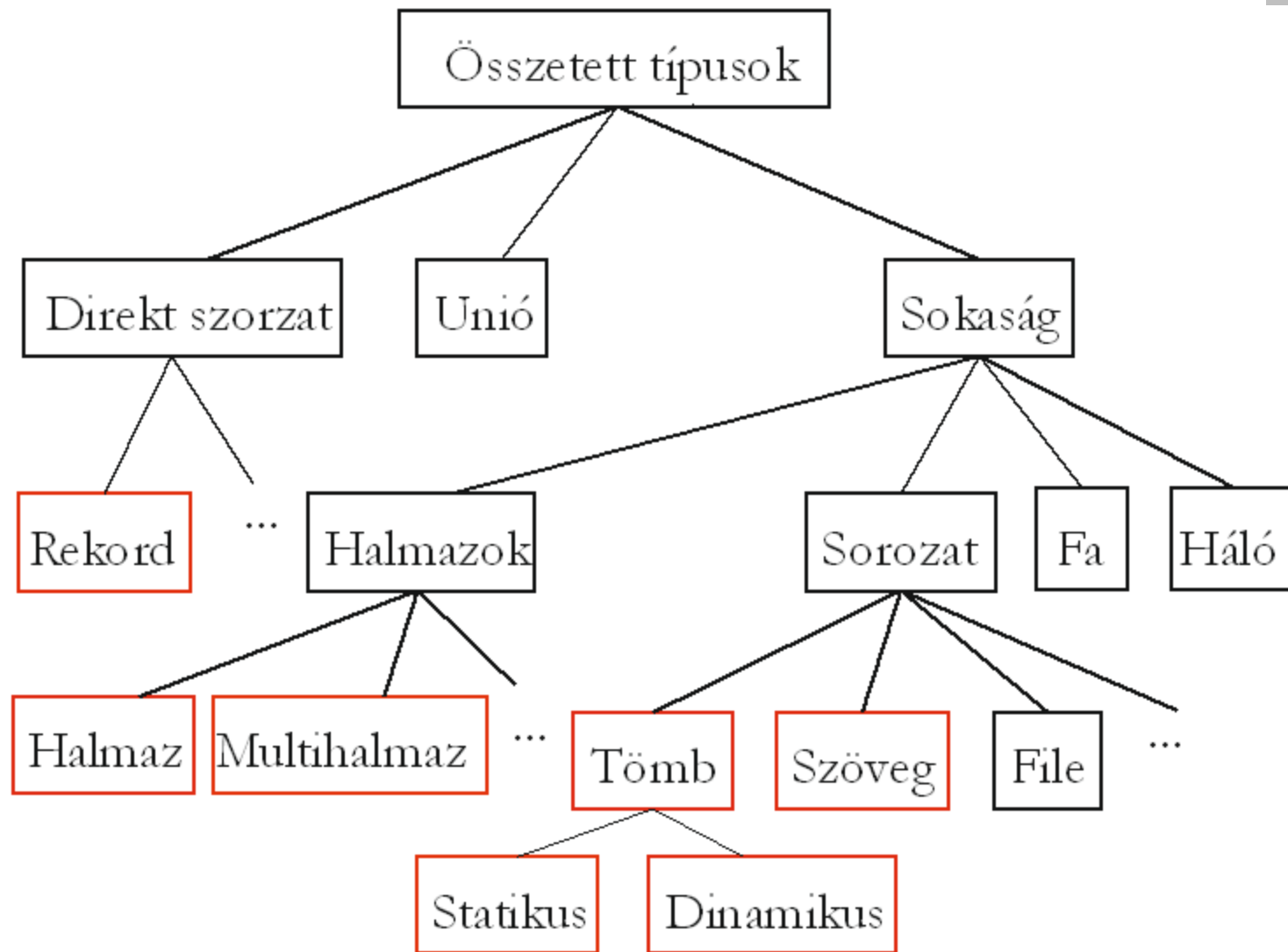
$Előbb := i < j$

Problémák: 1) Mi lenne, ha az előfeltétel nem teljesülne?

2) Lehetne-e $\text{"a"} \approx \text{"A"}$? $(i-1) \div 2 < (j-1) \div 2$



Összetett típusok



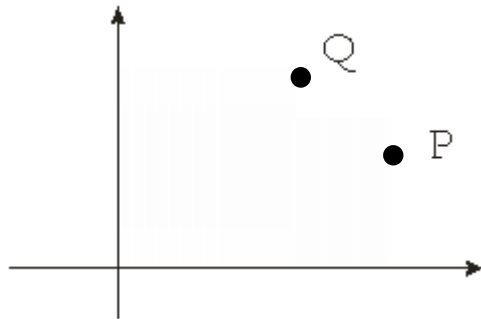
Függvények (irány)



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)

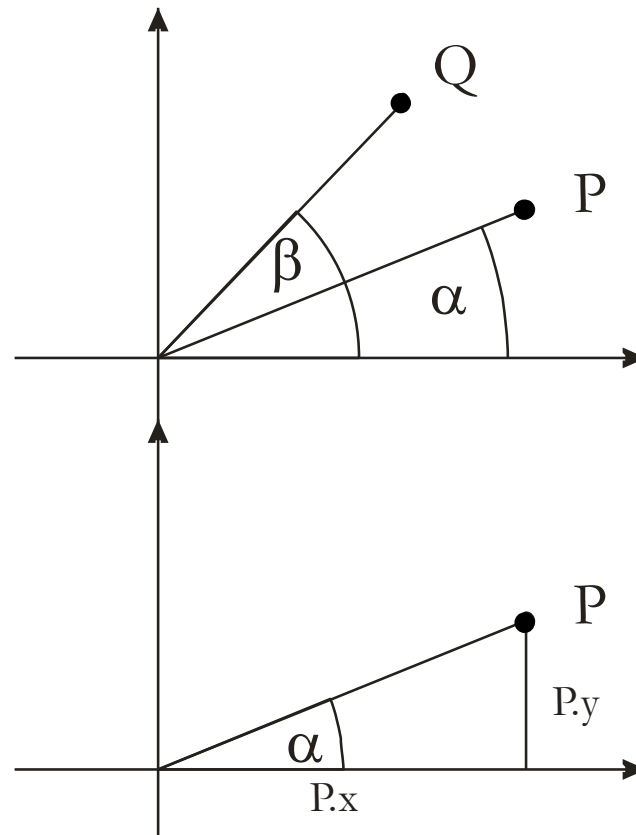


Értelmezés:

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

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

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



Függvények (irány)



$$\begin{aligned}\alpha < \beta &\Leftrightarrow \tan(\alpha) < \tan(\beta) \Leftrightarrow \\ &P.y/P.x < Q.y/Q.x \rightarrow \\ &P.y * Q.x < Q.y * P.x \Leftrightarrow \\ &P.y * Q.x - Q.y * P.x < 0\end{aligned}$$

Állítás:

$\text{Irány}(P, Q) = \text{sgn}(P.y * Q.x - Q.y * P.x)$
(és ez igaz **nem csak** az 1. síknegyedben).

Ellenőrizze a teljesülését:

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



Függvények (irány)

Specifikáció:

- Bemenet: $P, Q \in \text{Pont}$
 $\text{Pont} = x \times y, x, y = \mathbb{Z}$
 - Kimenet: $\text{Ir} \in \mathbb{Z}$
 - Előfeltétel: –
 - Utófeltétel: $\text{Ir} = \text{Irány}(P, Q)$
 - Definíció: $\text{Irány} : \text{Pont} \times \text{Pont} \rightarrow \mathbb{Z}$
- Függvény-szignatúra:
 $fv : \text{Honnan} \rightarrow \text{Hova}$
- $\text{Irány}(p, q) := \text{sgn}(p.y * q.x - q.y * p.x)$
- Aktuális paraméterek
- Formális paraméterek
- Aktuális paraméterek

Algoritmus:

$\text{Ir} := \text{Irány}(P, Q)$



Függvények (irány)



Az **Irány** függvény kiszámítását tekintsük **önálló feladatnak!**

Definíció: $\text{Irány}:\text{Pont}\times\text{Pont}\rightarrow\mathbb{Z}$
 $\text{Irány}(p,q):=\text{sgn}(p.y*q.x-q.y*p.x)$

Specifikáció:

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

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



Függvények (irány)



- Specifikáció → algoritmus
- függvénydefiníció:

Definíció: $\text{Irány}:\text{Pont}\times\text{Pont}\rightarrow\mathbb{Z}$
 $\text{Irány}(p,q):=\text{sgn}(p.y*q.x-q.y*p.x)$

$\text{Irány}(p,q:\text{TPont}):\text{Egész}$

Formális paraméterek

Változó
S:Egész
F:Egész

$\text{Irány}:=\text{sgn}(\text{p.y}*\text{q.x}-\text{q.y}*\text{p.x})$



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 } végzőjelig tart, kivéve azt a beágyazott blokkot, és ennek beágyazottjait, amelyben újra lett deklarálnva.

```
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:

Irány(p,q:TPont):Egész

			Változó S:Valós F:Egész
S:=p.y*q.x-q.y*p.x			
S<0	S=0	S>0	
F:=-1	F:=0	F:=1	
Irány:=F			

```
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!

A main előtt.

➤ Függvény-definíció:

A fejsor, a végén pontosvessző **nélkül**

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

void esetén fvÉrték nélküli return, vagy return nélkül!

A main után.



Függvények

C++ tudnivalók – összefoglalás:

➤ Formális skalár paraméter:

Pontosabban: **nem**
tömb

- bemeneti → **nincs** speciális kulcsszó
- kimeneti → **&** a speciális prefix „kulcsszó”

➤ Aktuális skalár paraméter:

Pontosabban: **nem**
tömb

ha a megfelelő formális paraméter

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

lehet.



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.:

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

Input-paraméterek.

- **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!

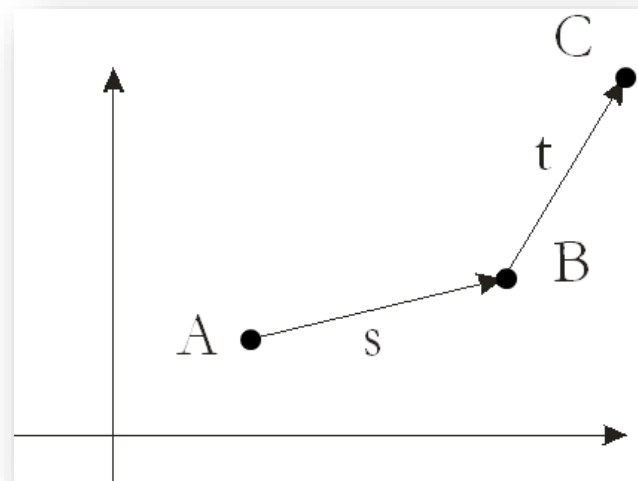


Függvények (fordul)



Feladat:

Egy s ($A \rightarrow B$) szakaszhoz képest egy t ($B \rightarrow C$) 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!

$$\text{Fordul}(A, B, C) = \text{Irány}(B - A, C - A)$$



Függvények (fordul)



Specifikáció:

➤ Bemenet: $A, B, C \in \text{Pont}$, $\text{Pont} = \dots$

➤ Kimenet: $\text{Ford} \in \mathbb{Z}$

➤ Előfeltétel: –

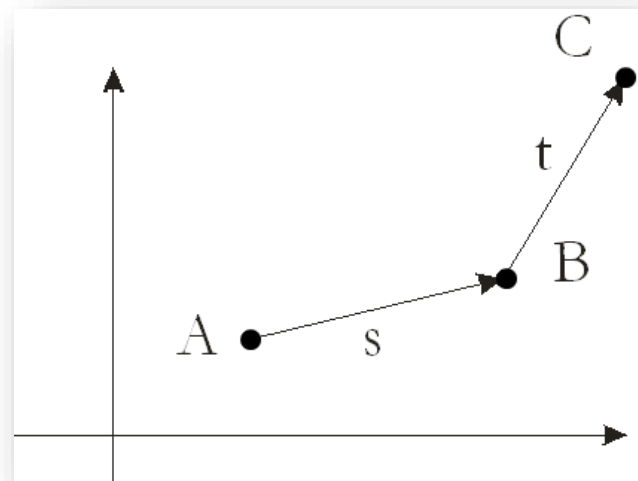
➤ Utófeltétel: $\text{Ford} = \text{Fordul}(A, B, C)$

➤ Definíció: $\text{Fordul}: \text{Pont}^3 \rightarrow \mathbb{Z}$

$$\text{Fordul}(a, b, c) := \text{Irány}(b - a, c - a)$$

$$- : \text{Pont}^2 \rightarrow \text{Pont}$$

$$P - Q := \text{Pont}(P.x - Q.x, P.y - Q.y)$$



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)

Algoritmus:

$\text{Ford} := \text{Fordul}(A, B, C)$

- A megoldásban hívjuk az Irány függvényt!

Finomítás₁:

$\text{Fordul}(a, b, c: \text{TPont}): \text{Egész}$

Változó
 $p, q: \text{TPont}$

$p.x := b.x - a.x$

$p.y := b.y - a.y$

$q.x := c.x - a.x$

$q.y := c.y - a.y$

$\text{Fordul} := \text{Irány}(p, q)$

Specifikáció:

- Bemenet: $A, B, C \in \text{Pont}$, $\text{Pont} = \dots$
- Kimenet: $\text{Ford} \in \mathbb{Z}$
- Előfeltétel: –
- Utófeltétel: $\text{Ford} = \text{Fordul}(A, B, C)$
- Definíció: $\text{Fordul}: \text{Pont}^3 \rightarrow \mathbb{Z}$
 $\text{Fordul}(a, b, c) := \text{Irány}(b-a, c-a)$

```
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;

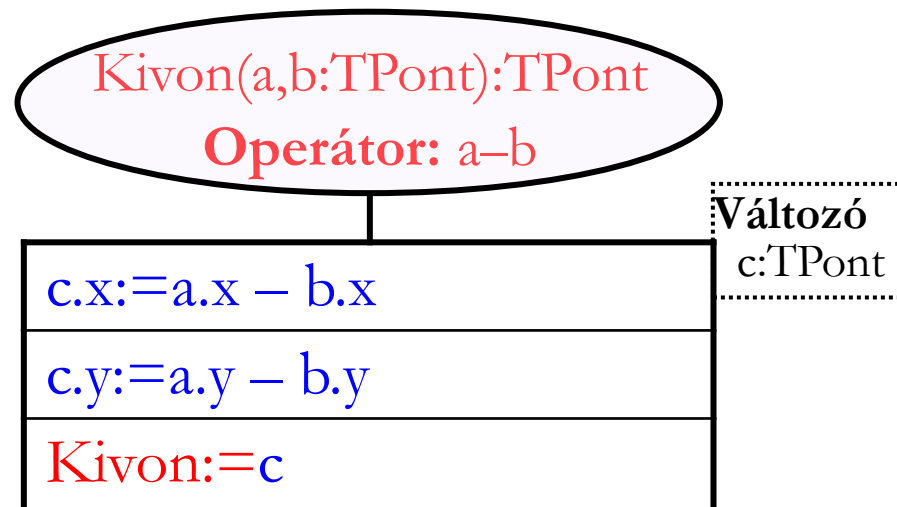
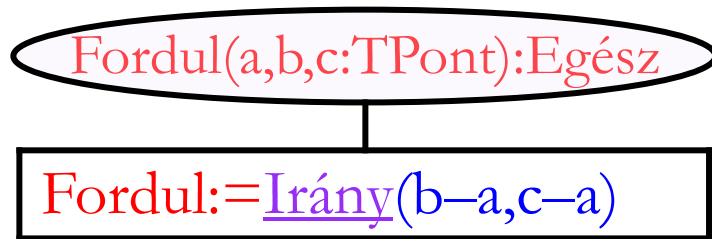
    return Irany(p, q);
}
```



Függvények (fordul)

- A Fordul-ban meghívjuk a TPont kivonás operátorát!

Finomítás₂:



```
int Fordul(TPont a, TPont b,
           TPont c)
{
    return Irany(b-a, c-a);
}
```

```
TPont operator -(TPont a, TPont b)
{
    TPont c;
    c.x=a.x - b.x;
    c.y=a.y - b.y;
    return c;
}
```



Függvények (rajta?)

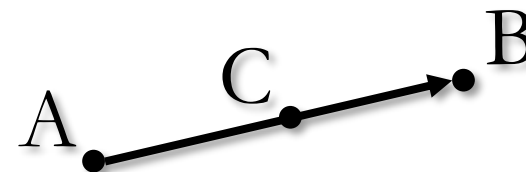


Feladat:

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

Specifikáció:

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



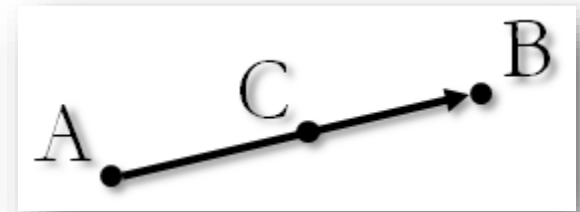
Függvények (rajta?)



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

$\text{Rajta: Pont}^3 \rightarrow \mathbb{L}$

$\text{Rajta}(a,b,c) := \text{Fordul}(a,c,b)=0$ és
 $\text{Közte}(a.x,c.x,b.x)$ és
 $\text{Közte}(a.y,c.y,b.y)$



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

$\text{Közte: } \mathbb{Z}^3 \rightarrow \mathbb{L}$

$\text{Közte}(r,s,t) := r \leq s \leq t \text{ vagy } t \leq s \leq r$



Függvények (rajta?)

Algoritmus:

$$\text{RajtaE} := \text{Rajta}(A, B, C)$$

➤ Finomítások:

$\text{Rajta}(a, b, c: \text{TPont}): \text{Logikai}$

$$\begin{aligned} \text{Rajta} := & \text{Fordul}(a, c, b) = 0 \text{ és} \\ & \text{Közte}(a.x, c.x, b.x) \text{ és} \\ & \text{Közte}(a.y, c.y, b.y) \end{aligned}$$

$\text{Közte}(r, s, t: \text{Egész}): \text{Logikai}$

$$\text{Közte} := r \leq s \text{ és } s \leq t \text{ vagy } t \leq s \text{ és } s \leq r$$

Specifikáció:

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

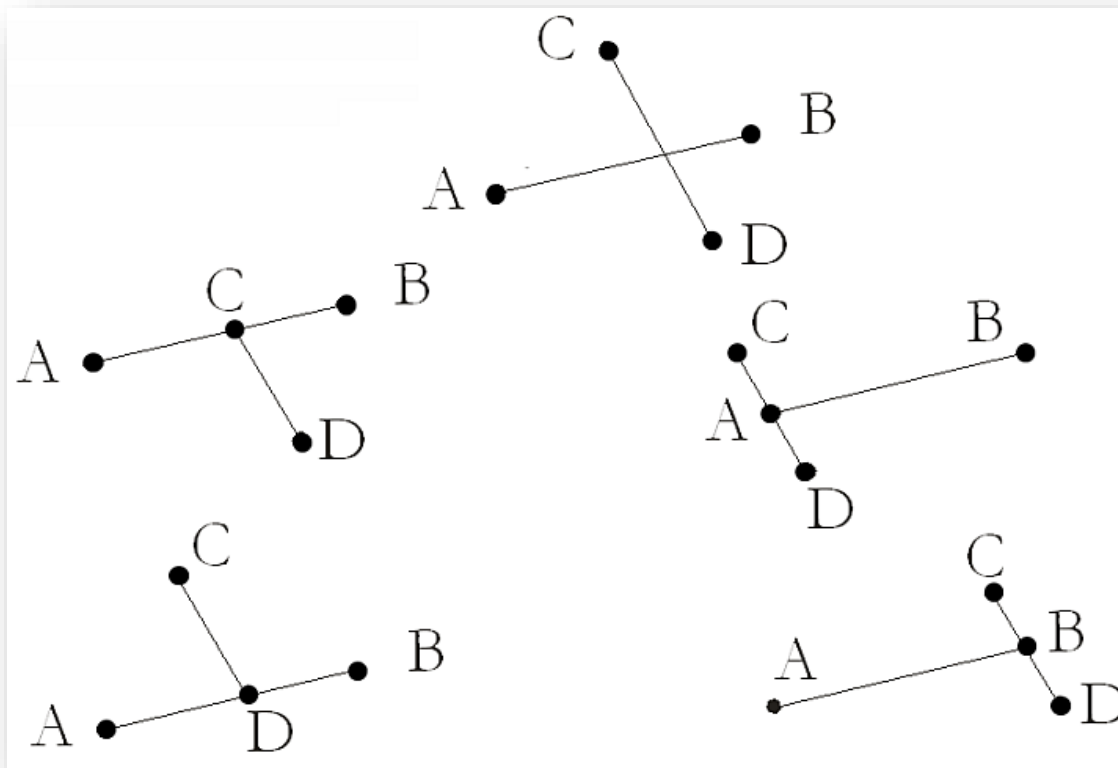


Függvények (metszi?)



Feladat:

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



Függvények (metszi?)



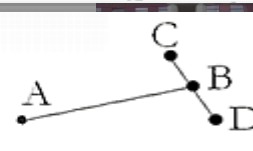
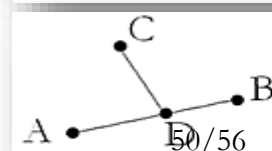
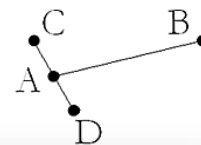
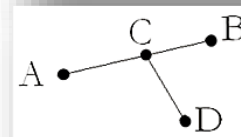
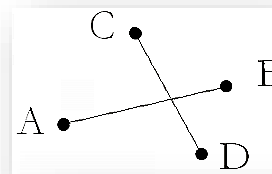
Specifikáció:

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

$\text{Metszi}: \text{Pont}^4 \rightarrow \mathbb{L}$

$\text{Metszi}(A, B, C, D) :=$

(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?)



Finomítás:

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)

Definíció:

Metszi: Pont⁴ → L

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 (háromszögben?)

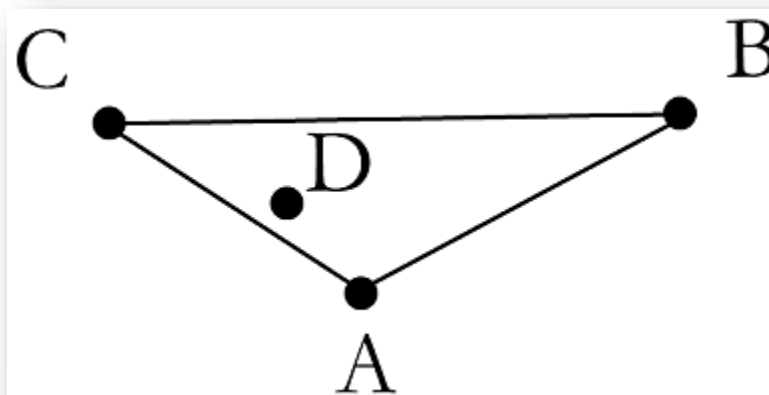


Feladat:

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

Megoldásötlet:

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



Függvények (háromszögben?)

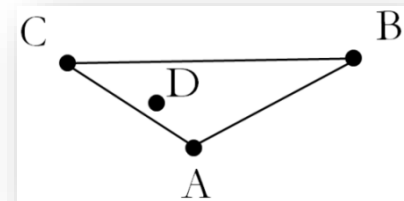


Specifikáció:

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

$\text{BelülE}: \text{Pont}^4 \rightarrow \mathbb{L}$

$\text{BelülE}(a, b, c, d) := \text{Fordul}(a, b, d) = \text{Fordul}(b, c, d)$
és $\text{Fordul}(b, c, d) = \text{Fordul}(c, a, d)$



Függvények (háromszögben?)



Finomítás:

➤ Definíció:

$\text{BelülE}:\text{Pont}^4 \rightarrow \mathbf{L}$

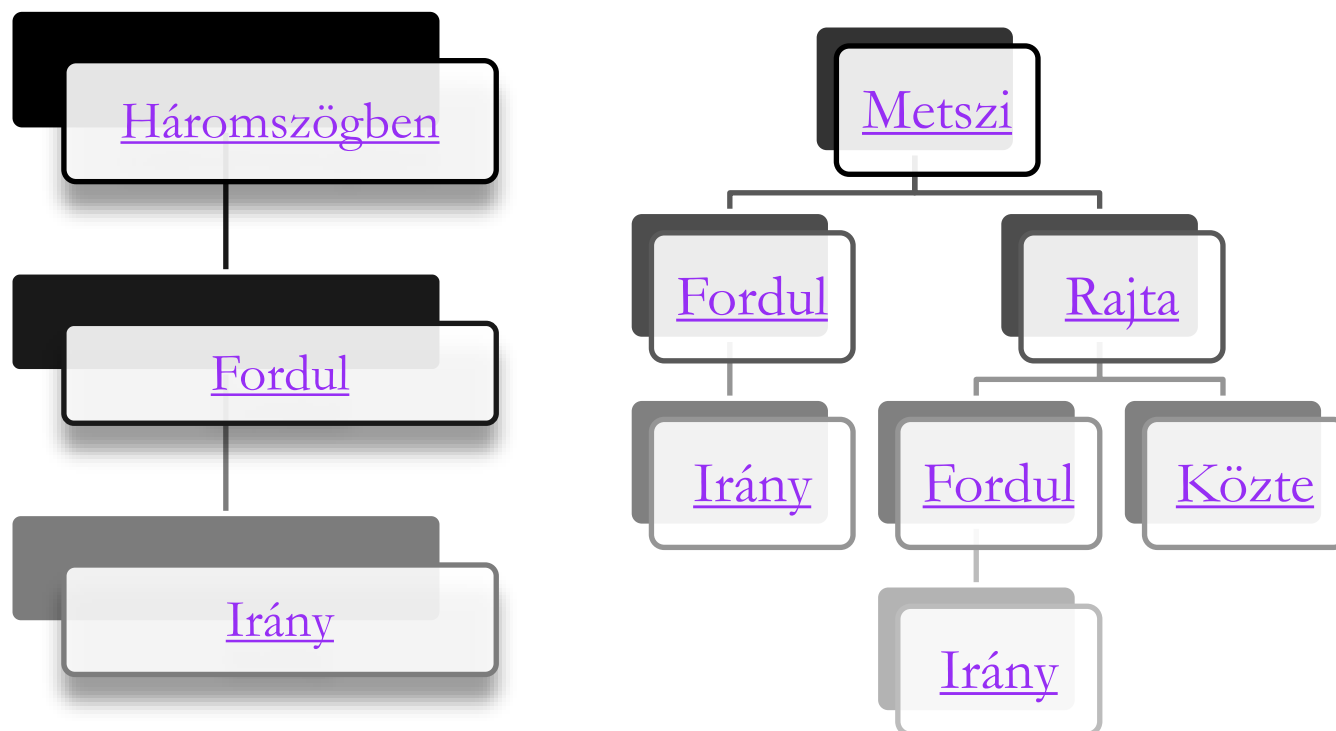
$\text{BelülE}(a,b,c,d) := \text{Fordul}(a,b,d) = \text{Fordul}(b,c,d)$
és $\text{Fordul}(b,c,d) = \text{Fordul}(c,a,d)$

$\text{BelülE}(a,b,c,d:\text{TPont}):\mathbf{Logikai}$

$\text{BelülE} := \text{Fordul}(a,b,d) = \text{Fordul}(b,c,d)$
és $\text{Fordul}(b,c,d) = \text{Fordul}(c,a,d)$



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



Tartalom



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

