

Verem, Sor

3. előadás

Verem

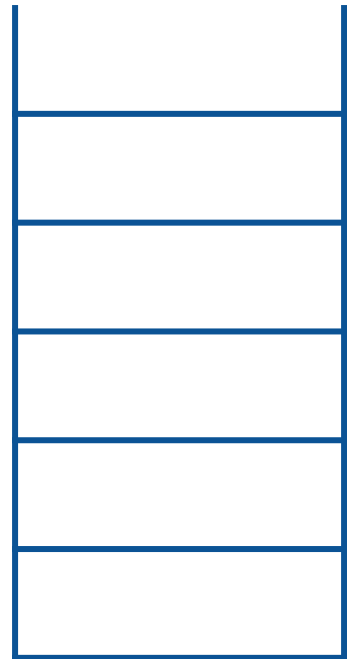
Stack, LIFO

Példa: VEREM

- LIFO: last-in, first-out
- Köznapi fogalma



- Kimászás sorrendje



A verem ADT axiomatikus leírása

E alaptípus feletti

V verem típus jellemzése

- Műveletek

- $\text{empty} \rightarrow V$
- $\text{isempty} \quad V \rightarrow L$
- $\text{push} \quad V \times E \rightarrow V$
- $\text{pop} \quad V \rightarrow V \times E$
- $\text{top} \quad V \rightarrow E$

- Megszorítások:

- pop és top értelmezési tartománya
 - $V \setminus \{\text{empty}\}$

- Műveletek jelentése

- Üres verem létrehozása
- Üres a verem?
- Elem betétele a verembe
- Elem kivétele a veremből
- Felső elem lekérdezése

- Figyelem!

- A műveletek között nem szerepel „isfull” művelet!

A verem ADT axiomatikus leírása

- Axiómák:
 - $\text{isempty}(\text{empty})$
 - Vagy: $v = \text{empty} \rightarrow \text{isempty}(v)$
 - $\text{isempty}(v) \rightarrow v = \text{empty}$
 - $\neg \text{isempty}(\text{push}(v, e))$
 - $\text{pop}(\text{push}(v, e)) = (v, e)$
 - $\text{push}(\text{pop}(v)) = v$
 - $\text{top}(\text{push}(v, e)) = e$

A verem ADT funkcionális leírása

- Matematikai reprezentáció
 - a verem rendezett párok halmaza
$$v = \{(e_1, t_1), \dots (e_i, t_i) \mid a t_j \text{ komponensek különbözőek}\}$$
 1. komponens: a veremben elhelyezett (push) érték
 2. komponens: a verembe helyezés (push) időpontja
- Megszorítás (invariáns)
 - az idő értékek különbözők
- A valóságban nem így implementáljuk!

A verem ADT funkcionális leírása

- A „pop” specifikációja:
 - Állapottér
 - $V \times E$ (Állapottér típusai: $V = \{(e_i, t_i), \dots\}$)
 - $v \quad e$ (Állapottér változói)
 - Paramétertér
 - V és v'
 - Előfeltétel és utófeltétel:
 - $Ef = (v = v' \wedge v' \neq \emptyset)$
 - $Uf = \left((v = v' \setminus \{(e_j, t_j)\}) \wedge (e = e_j) \wedge ((e_j, t_j) \in v') \wedge \right.$

Verem

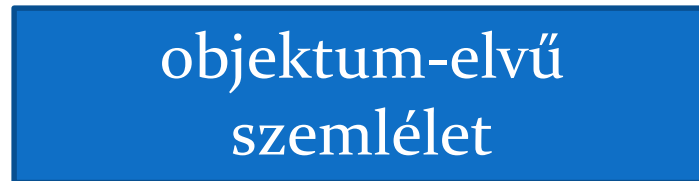
Műveletek jelölése:

`push(v, e)`

`v.push(e)`

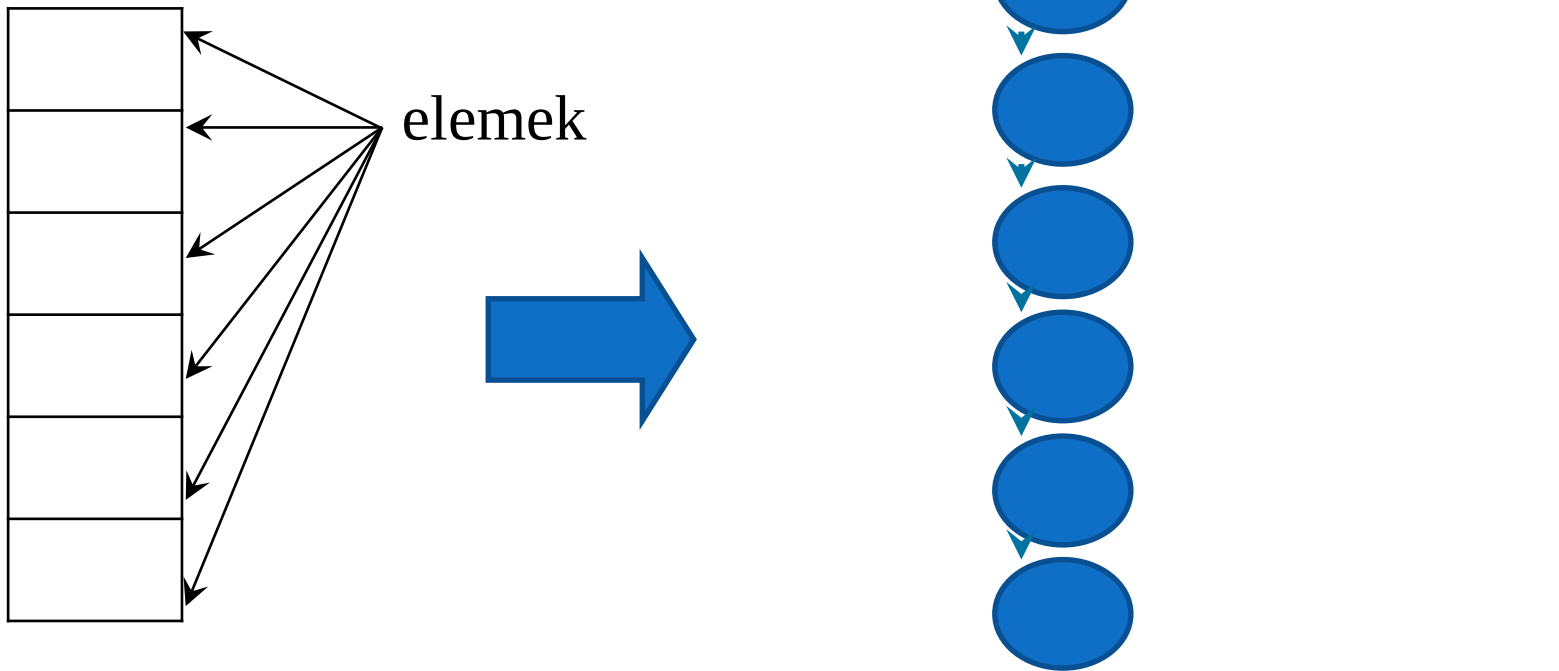
procedurális
szemlélet

objektum-elvű
szemlélet



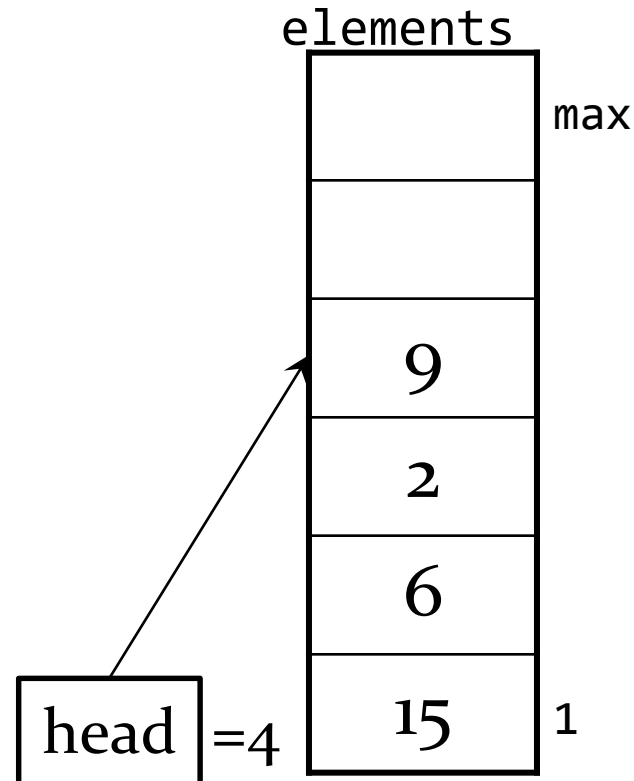
Verem – ADS

- Lineáris adatszerkezet



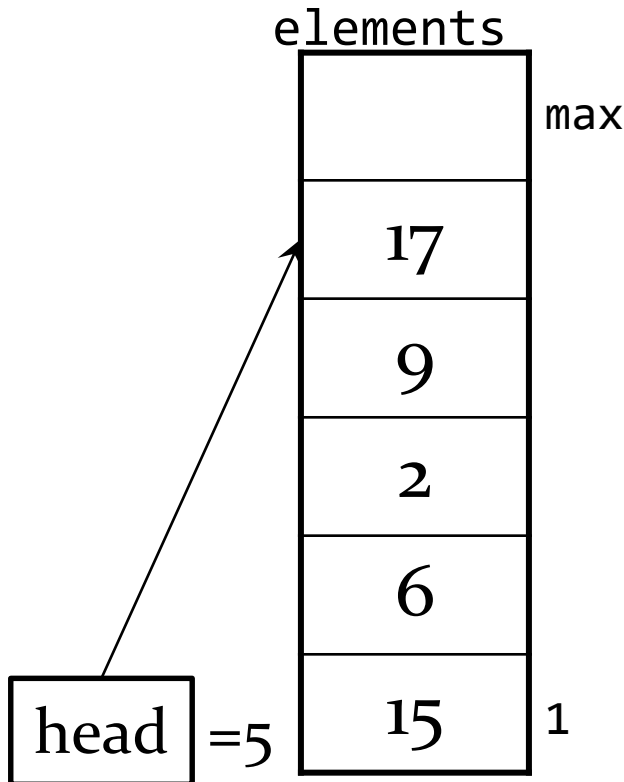
Reprezentáció

- Aritmetikai ábrázolás:
 - egy max hosszú vektor
(ez az elemek tömbje)
`elements[1..max]`
 - a verem tetejének mutatója
 $\text{head} \in [0, \text{max}]$
 $\text{head}=0 \Leftrightarrow$ üres a verem
 - Választási lehetőség, hogy hova mutat a head
 - Az első szabad helyre
 - Az utolsó elfoglalt helyre

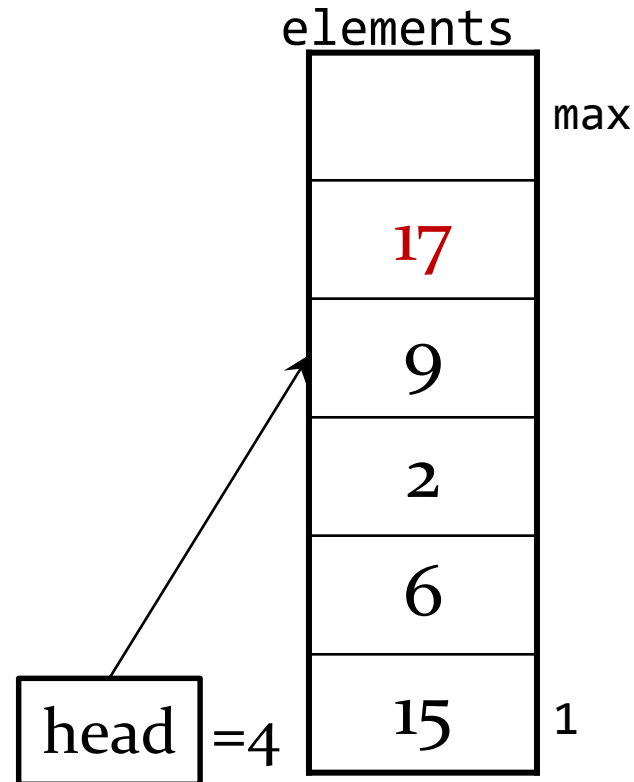


Reprezentáció

v.push(17) után



v.pop() után



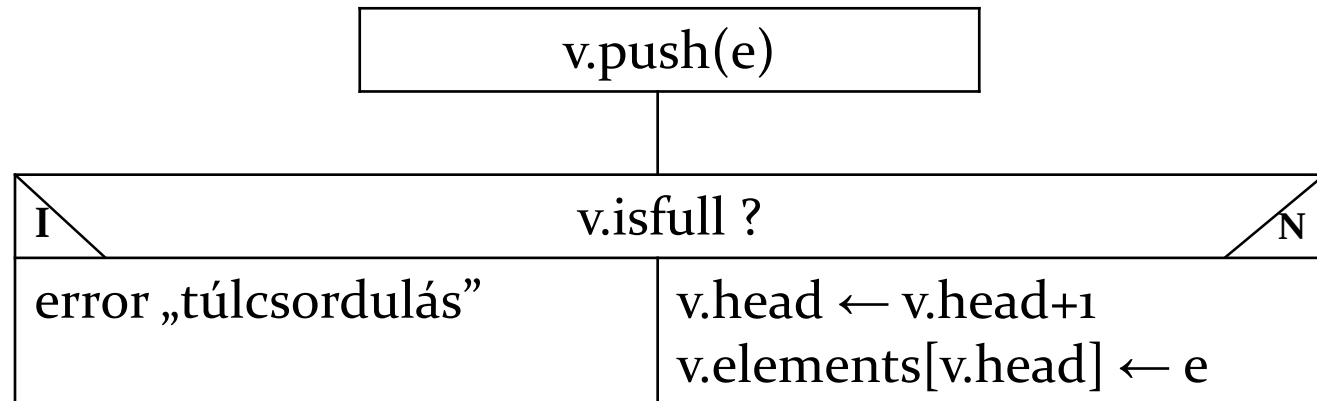
Egyenlő a két verem?

Implementáció

- Műveletek pszeudokódja/struktogramja:
 - `v.empty`
 - üresre állítja a vermet
 - `v.head ← 0`
 - `v.isempty`
 - Üres a verem? – Logikai értéket ad vissza
 - `return (v.head=0)`
 - `v.isfull`
 - Tele van a verem? – Logikai értéket ad vissza
 - `return (v.head=max)`

Implementáció

- `v.push(e)`
 - `e`-t beteszi a `v` verem tetejére
 - if `v.isfull`
 - then error „túlcsordulás”
 - else `v.head` $\leftarrow$ `v.head` +1
 - `v.elements[v.head]` $\leftarrow$ `e`
 - end if



Implementáció

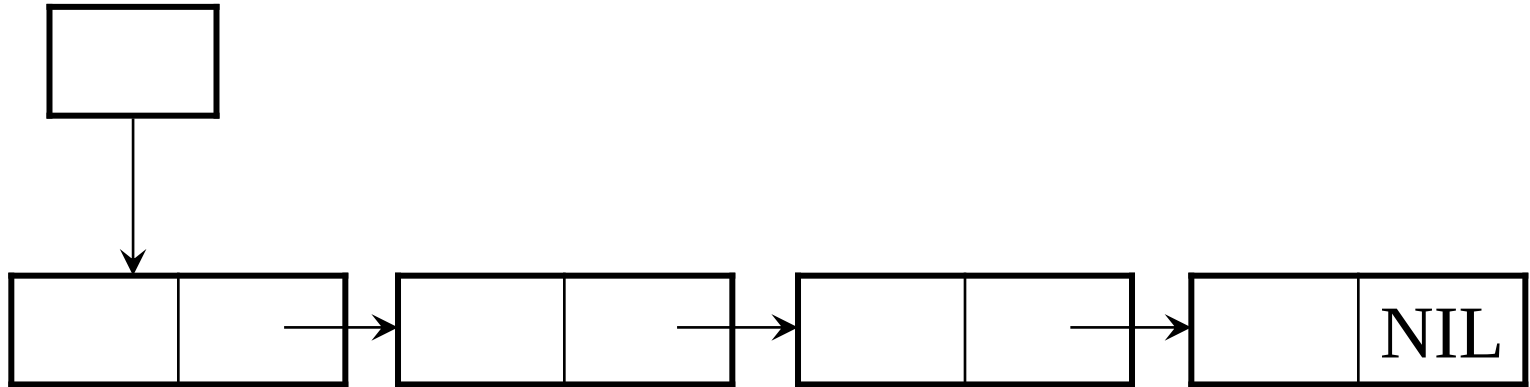
- `v.pop`
 - kiveszi a legfelső elemet és visszaadja
 - if `v isempty`
 - then error „alulcsordulás”
 - else `v.head ← v.head - 1`
 - return `v.elements[v.head+1]`
 - end if

Implementáció

- `v.top`
 - lekérdezi a legfelső elemet
 - `if v.isempty`
 - `then error „alulcsordulás”`
 - `else return v.elements[v.head]`
 - `end if`

Reprezentáció

- Lesz még láncolt ábrázolás is! (Gyakorlaton)



Sor

Queue, FIFO

Sor (Queue)

- FIFO: First-in, First-out
- Köznapi fogalma



A sor ADT axiomatikus leírása

E alaptípus feletti S sor típus jellemzése

- Műveletek

- $\text{empty} \rightarrow S$
- $\text{isempty} \quad S \rightarrow L$
- $\text{in} \quad S \times E \rightarrow S$
- $\text{out} \quad S \rightarrow S \times E$
- $\text{first} \quad S \rightarrow E$

- Megszorítások:

- out és first értelmezési tartománya
 - $S \setminus \{\text{empty}\}$

- Műveletek jelentése

- Üres sor létrehozása
- Üres a sor?
- Elem betétele a sorba
- Elem kivétele a sorból
- Első elem lekérdezése

- Figyelem!

- A műveletek között nem szerepel „isfull” művelet!

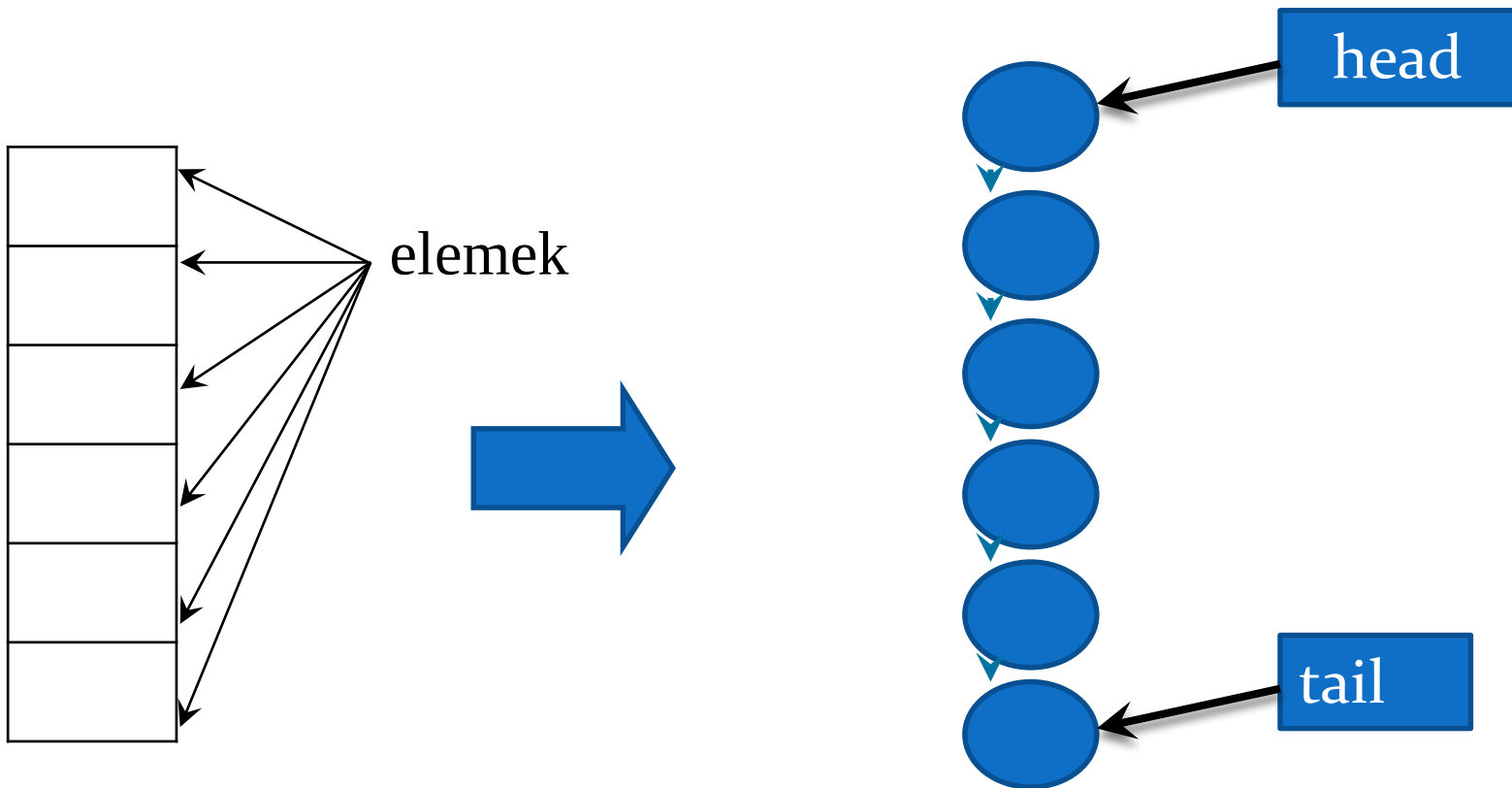
Sor – ADT

- Axiómák:

- $s = \text{empty} \rightarrow \text{isempty}(s)$
- $\text{isempty}(s) \rightarrow s = \text{empty}$
- $\neg \text{isempty}(\text{in}(s, e))$
- $\neg \text{isempty}(s) \rightarrow \text{out}(\text{in}(s, e))_2 = \text{out}(s)_2$
- $\neg \text{isempty}(s) \rightarrow \text{in}(\text{out}(s)_1, e) = \text{out}(\text{in}(s, e))_1$
- $\text{isempty}(s) \rightarrow \text{out}(\text{in}(s, e))_2 = e$
- $\text{first}(s) = \text{out}(s)_2$
- Ahol: $(s, e)_1 = s$, $(s, e)_2 = e$
- Végig kell gondolni, önállóan – vizsgakérdés!

Sor – ADS

- Lineáris adatszerkezet

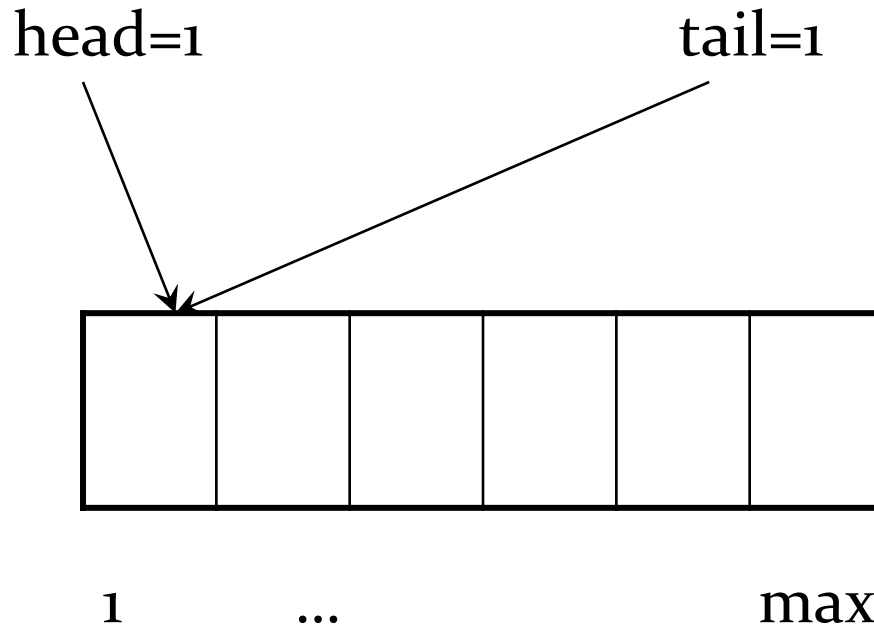


Reprezentáció

- Aritmetikai ábrázolás:
 - egy max hosszú vektor
 - ez az elemek tömbje
 - `elements[1..max]`
 - és a sor első elemének mutatója
 - $\text{head} \in [1, \text{max}]$
 - és a sor első üres (utolsó) helyének mutatója
 - $\text{tail} \in [1, \text{max}]$
- Vegyük észre, hogy az aritmetikai ábrázolás három részből áll!

Reprezentáció

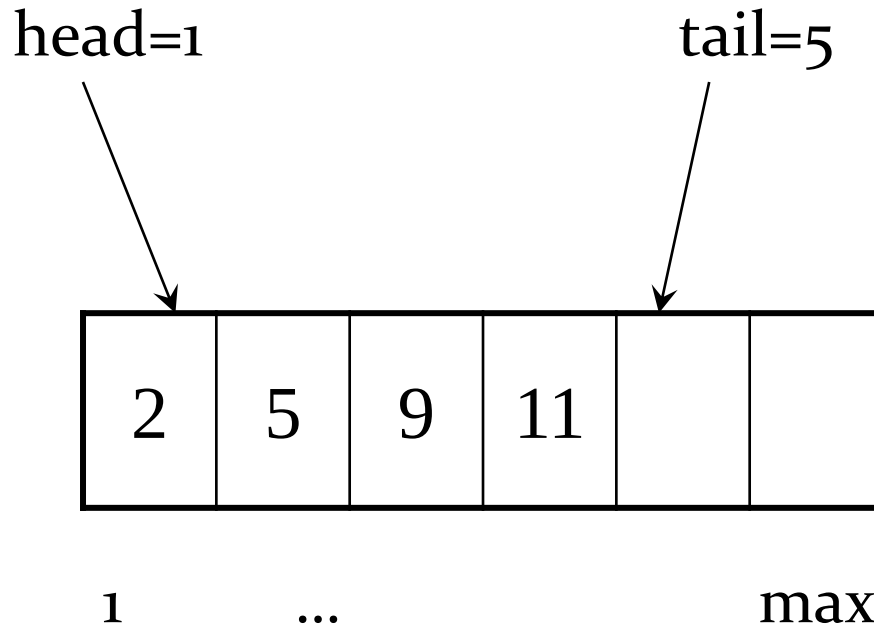
- Ciklikus ábrázolással
 - Kezdetben



Üres a sor

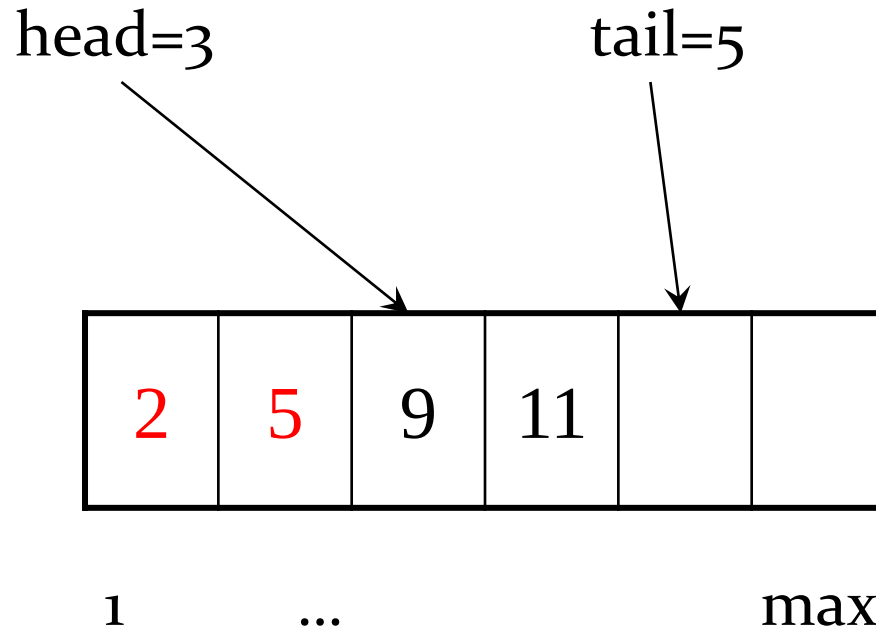
Reprezentáció

- 2, 5, 9, 11 betétele után:



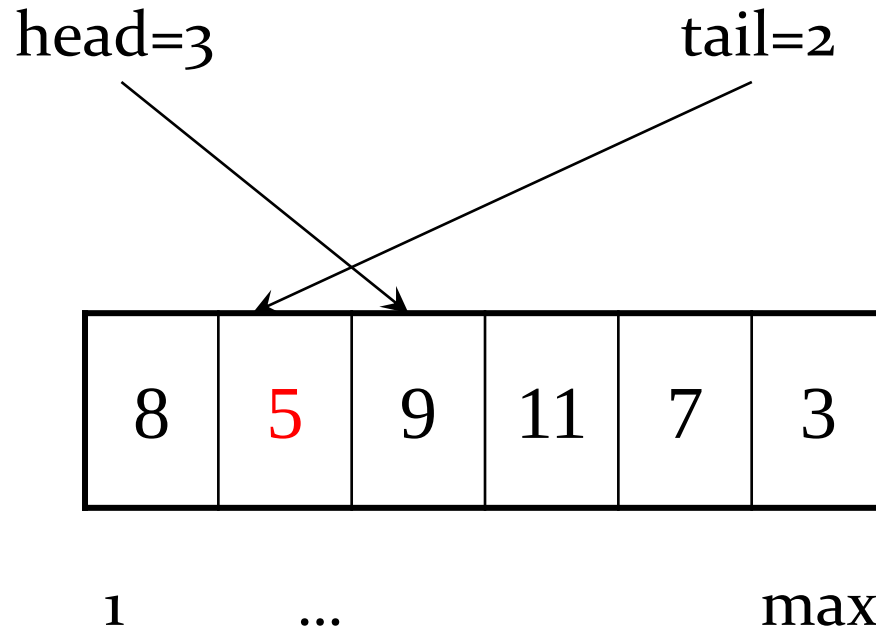
Reprezentáció

- 2, 5 kivétele után:



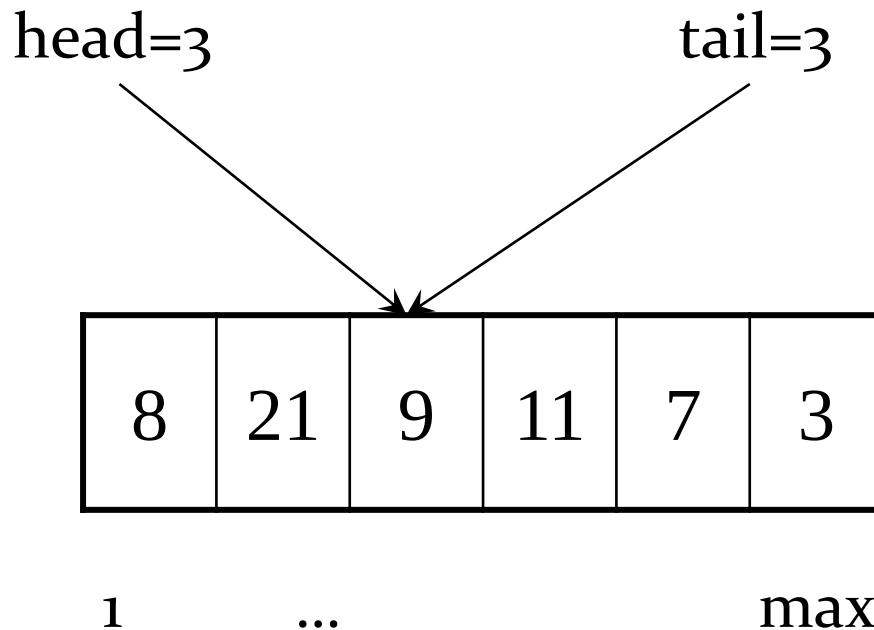
Reprezentáció

- 7, 3, 8 betétele után:



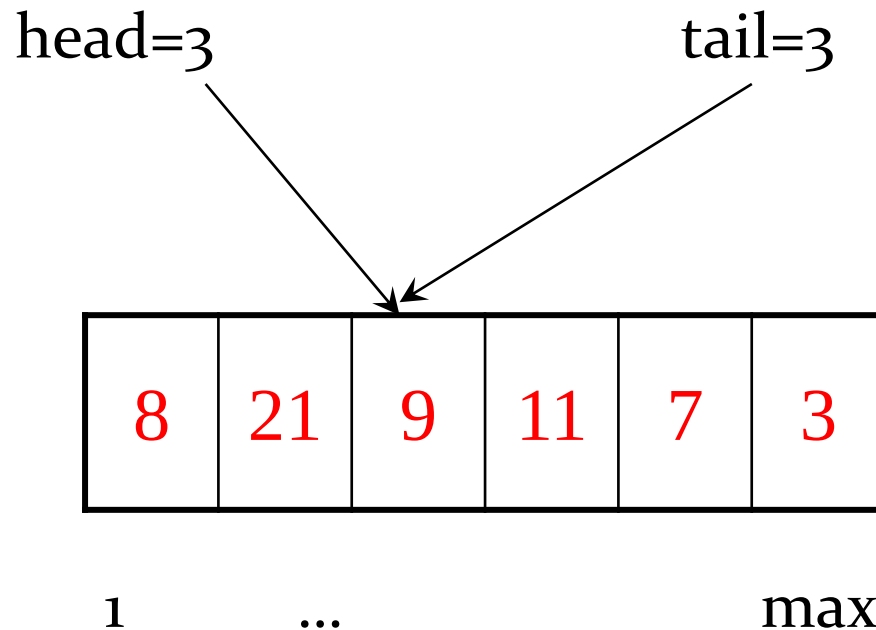
Reprezentáció

- 21 betétele után:



Reprezentáció

- 9, 11, 7, 3, 8, 21 kivétele után:



Üres a sor

Reprezentáció

- Mit tegyünk? – Milyen lehetőségek vannak:
 - Vezessünk be még egy jelzőt a reprezentációba, ami mutatja, hogy a sor üres-e
 - `empt` – kezdetben igaz, később vizsgáljuk, és megfelelően állítjuk
 - Vezessünk be még egy attribútumot a reprezentációba, ami mutatja, hogy hány elem van a sorban - `count`

Implementáció

- Műveletek pszeudokódja:
 - `s.empty`
 - üresre állítja a sort
 - `s.head ← 1; s.tail ← 1; s.empty ← true`
 - `s.isempty`
 - üres a sor? - logikai értéket ad vissza
 - `return s.empty`
 - `s.isfull`
 - tele van a sor?
 - `return ((not s.empty) and (s.head = s.tail))`

Implementáció

- `s.In(e)`
 - e-t beteszi az s sor végére
 - s.tail-t ciklikusan növeli
 - if `s.IsFull`
 - then error „túlcsordulás”
 - else `s.empty` $\leftarrow$ false
 - `s.elements[s.tail]` $\leftarrow$ e
 - if `s.tail=max`
 - then `s.tail` $\leftarrow$ 1
 - else `s.tail` $\leftarrow$ `s.tail+1`
 - end if
 - end if

Implementáció

- `s.Out`

```
-- kiveszi és visszaadja az s sor első elemét
-- s.head-et ciklikusan növeli
-- figyel, hogy nem üres-e a sor
if s.empt
  then error „alulcsordulás”;
  else e ← s.elements[s.head]
    if s.head=max
      then s.head ← 1
      else s.head ← s.head+1
    end if
    if s.head=s.tail then s.empt ← true end if
    return e
end if
```

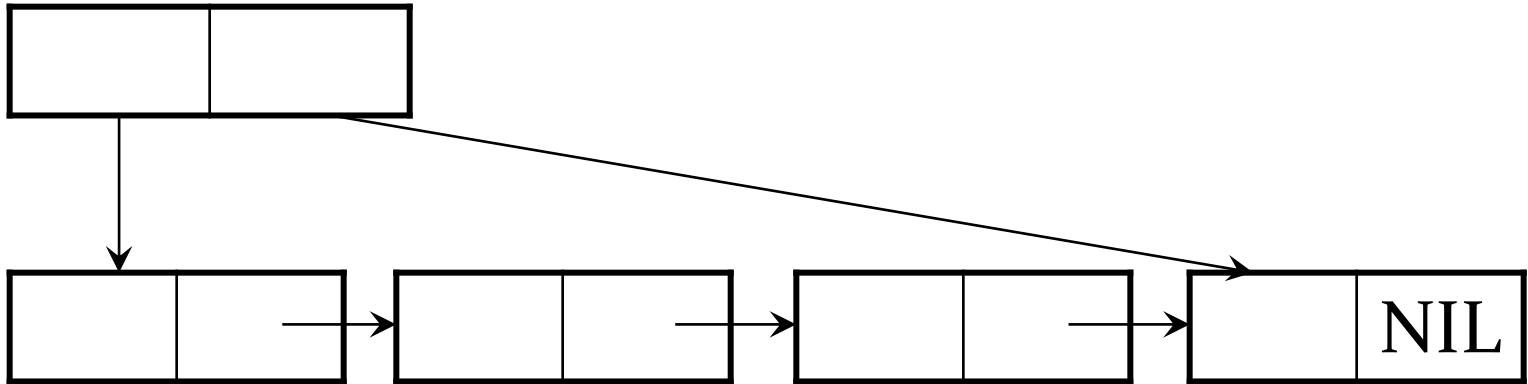
Implementáció

- `s.First`
 - visszaadja az `s` sor első elemét,
 - figyelj, hogy nem üres-e a sor

```
if s.empty
  then error „alulcsordulás”
  else return s.elements[s.head]
end if
```
- Lehetne az is, hogy a darabszámot tároljuk
 - Házi feladat: átgondolni

Reprezentáció

- Lesz még láncolt ábrázolás is!



Implementáció

- Vigyázni kell, amikor a programokat a választott programnyelven megvalósítjuk!
 - Például
 - C++ nyelv esetén a vektorok indexelése nullával kezdődik!
 - Értékadás jele
 - Egyenlőség vizsgálat jele

Lengyel forma

Az előzőek alkalmazása

Kifejezések

- Infix
 - Amikor az operátor az operandusok között van
 - $a + b$
- Postfix
 - Amikor az operátor az operandusok mögött van
 - $a b +$
- Prefix
 - Amikor az operátor az operandusok előtt van
 - $+a b$

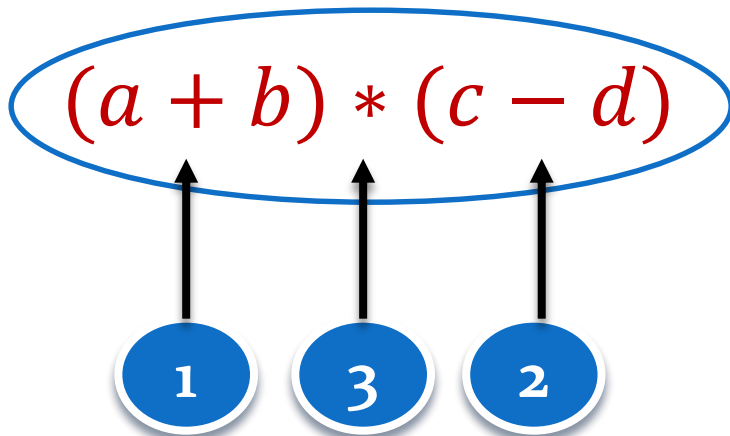
Lengyel forma

- J. Lukaszewitz lengyel matematikus használta először
 - Prefix forma
 - **Postfix forma – fordított lengyel forma**
- Átalakítás
 - Infix kifejezés
 - $(a + b) * (c - d)$
 - Postfix kifejezés
 - $a b + c d - *$
- Az Infix kifejezés átalakítható Postfix kifejezéssé
 - A kifejezések feldolgozásánál két fontos előnye van

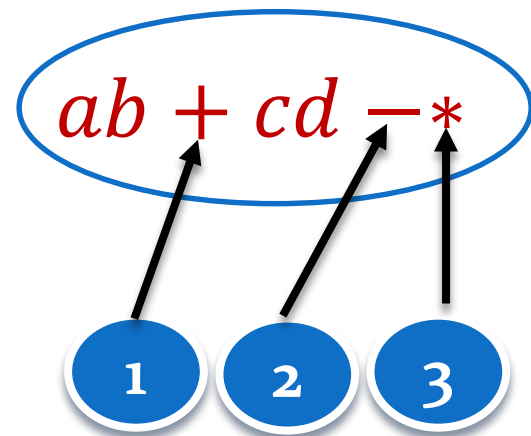
Fordított lengyel forma – előnyök

- A műveleti jelek olyan sorrendben követik egymást, amilyen sorrendben végre kell hajtani azokat.

Infix forma

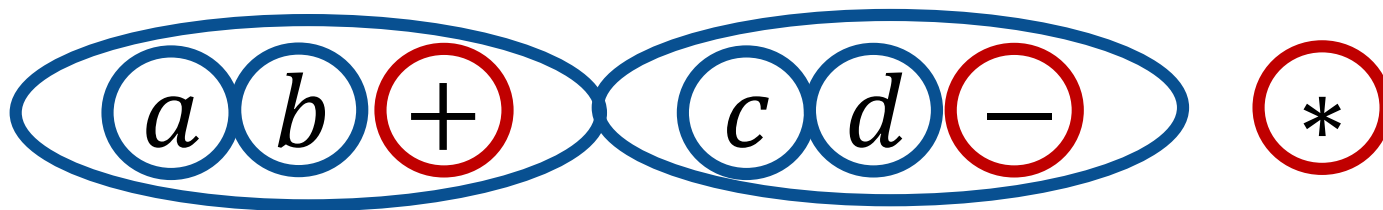


Lengyelforma



Fordított lengyel forma – előnyök

- A műveleti jel (operátor) közvetlenül az operandusai után áll
 - Infix forma: $(a + b) * (c - d)$
 - Postfix forma
 - Operátor
 - Operandus



További példák

- Kifejezés lengyel formára hozása
- Szokásos forma
 - $(1 + 2) * (3 - 4)$
 - $(a + b * c) * (d * 3 - 4)$
- Fordított lengyel forma
 - $1\ 2\ +\ 3\ 4\ -\ *$
 - $abc\ * \ +d\ 3\ * \ 4\ -\ *$

X:

(	1	+	2	)	*	(	3	-	4	)
---	---	---	---	---	---	---	---	---	---	---

sorozat



Y:

--	--	--	--	--	--	--	--	--	--	--

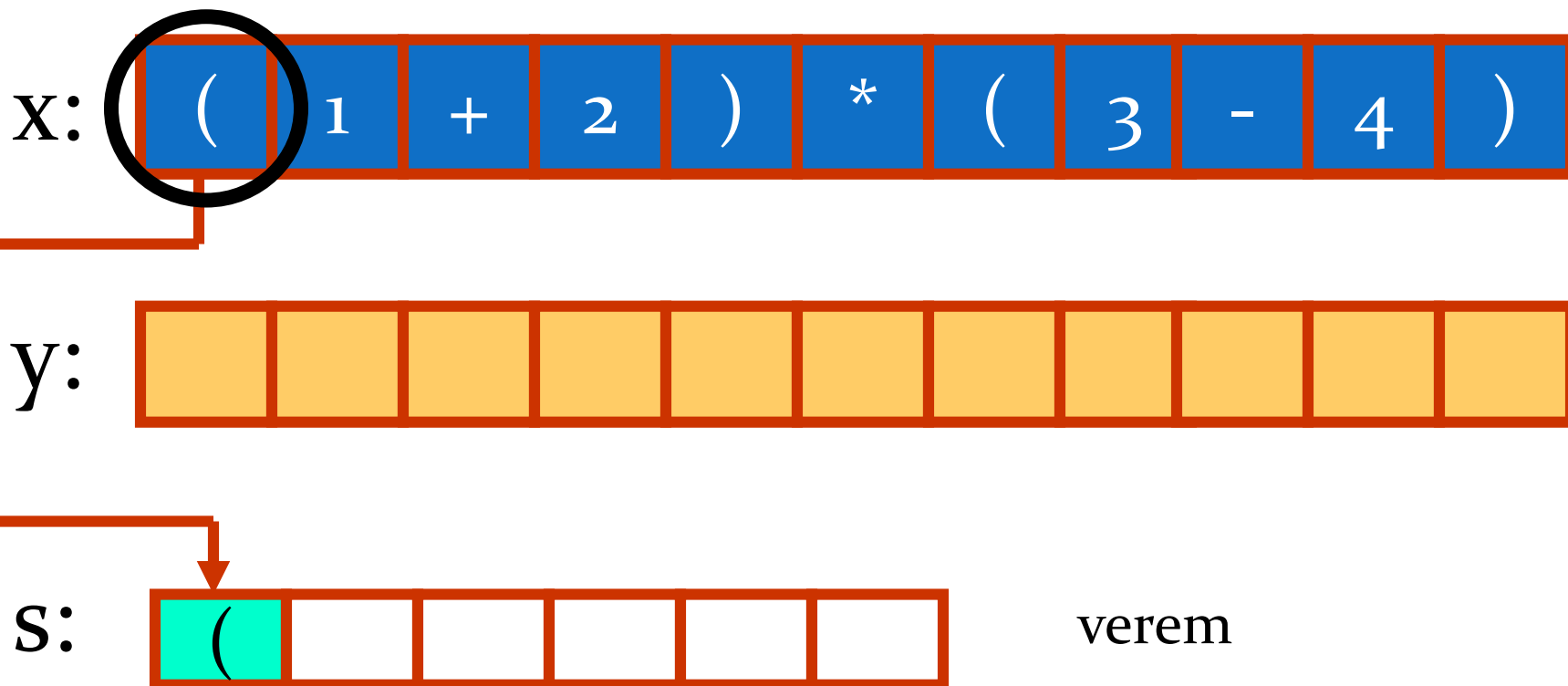
sorozat

S:

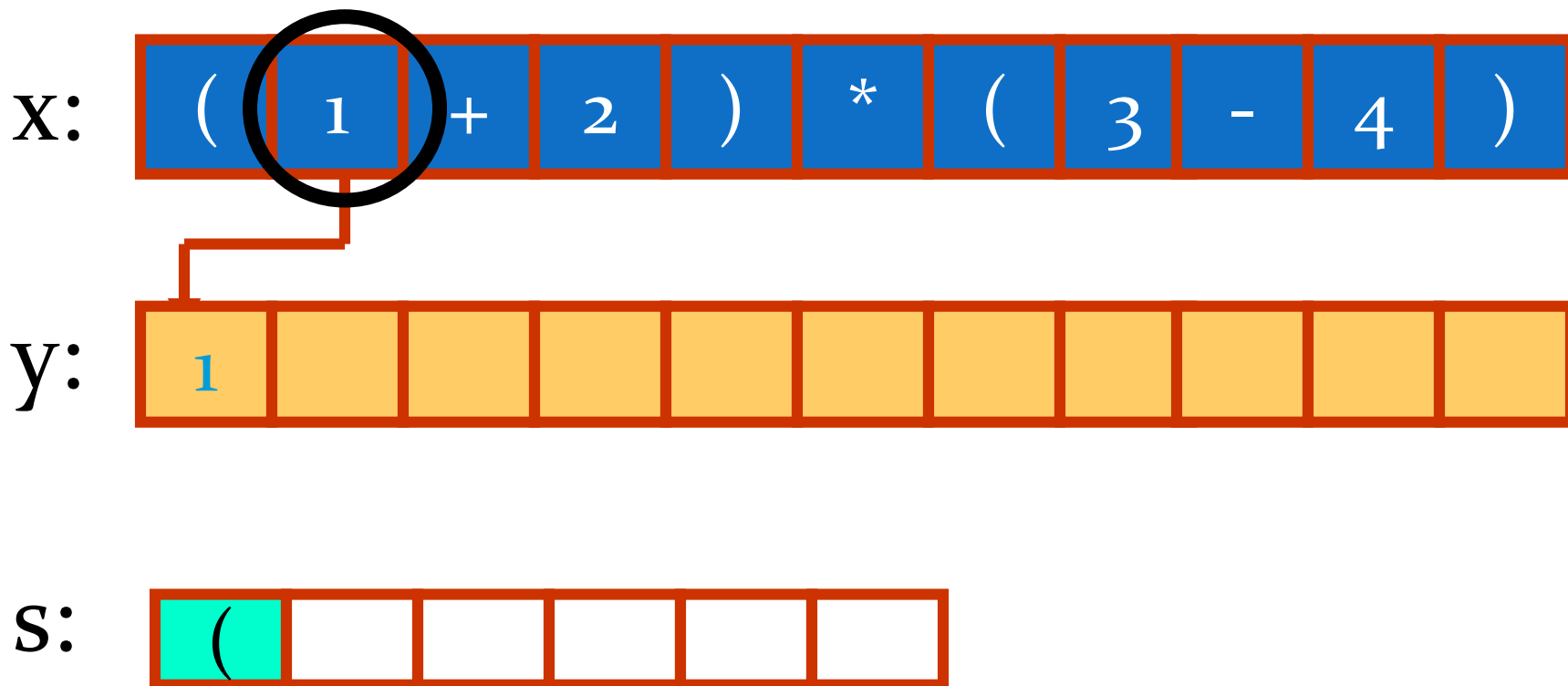
--	--	--	--	--	--

verem

Az x sortozatot balról jobbra haladva dolgozzuk fel.
A sortozat végét ; jelzi.



A következő szimbólum **nyitózárójel**: **tegyük a verembe.**



A következő szimbólum operandus: írjuk ki.

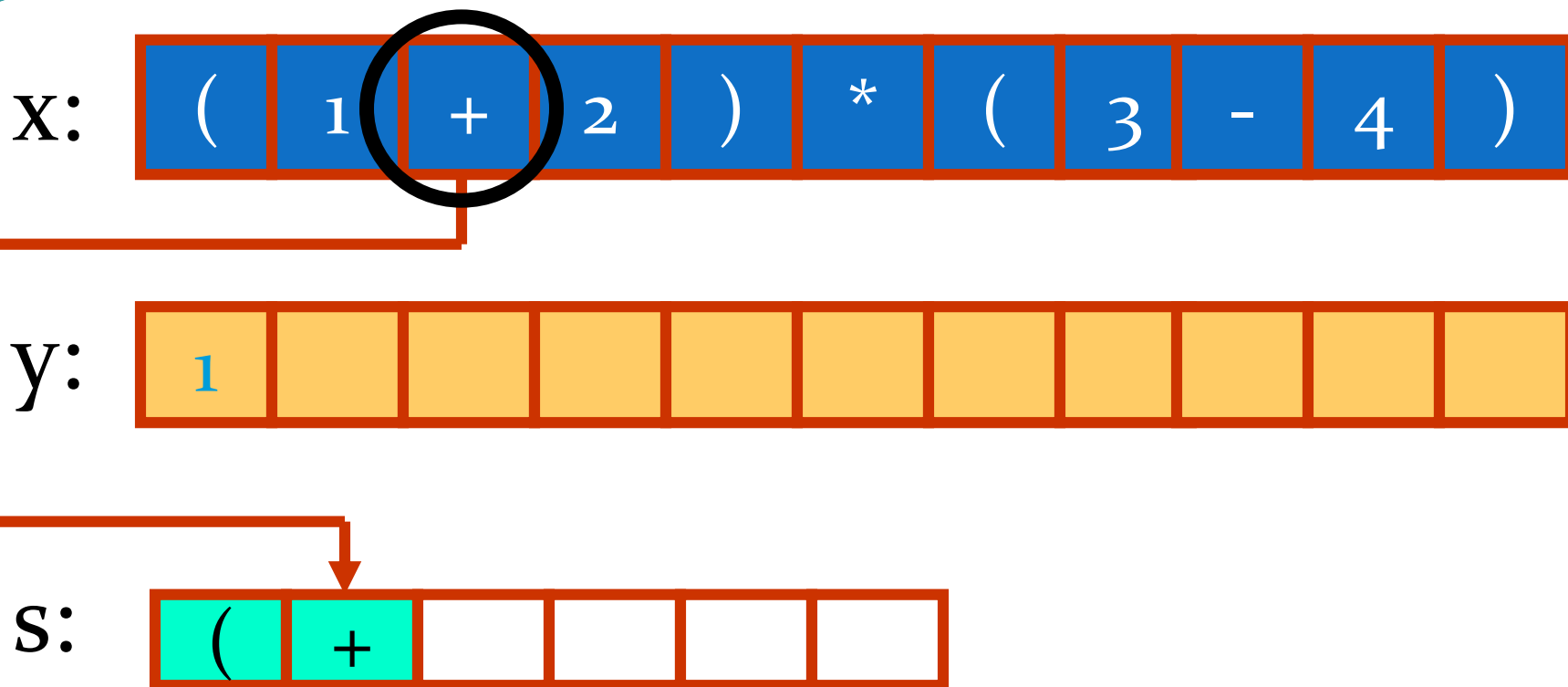
x: (1 + 2) * (3 - 4)

y: 1

s: (

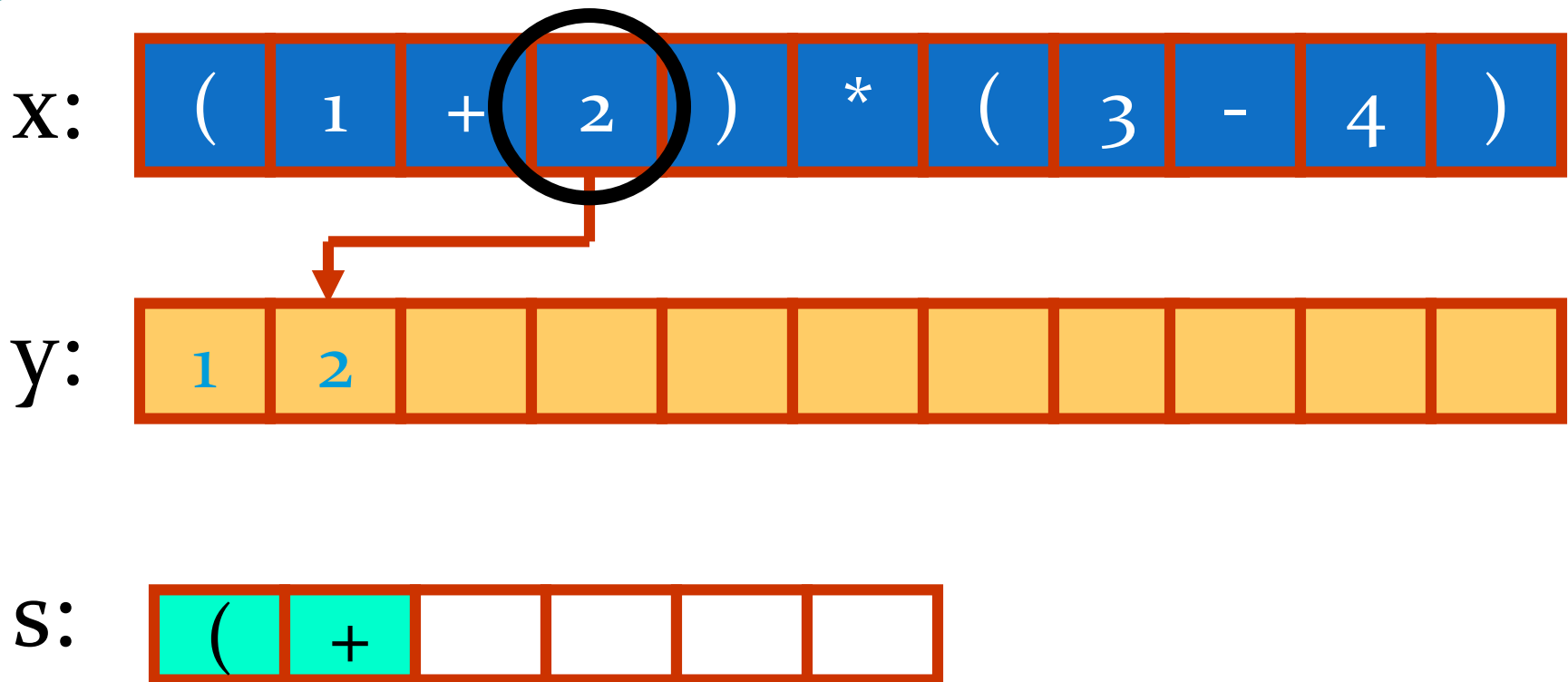
A következő szimbólum **operátor**:

1. **legfeljebb a nyitózárójelig** kivesszük a veremből az operátornál **nagyobb prioritású** operátorokat és **kiírjuk** azokat,



A következő szimbólum **operátor**:

1. legfeljebb a nyitózárójelig kivesszük a veremből az operátornál nagyobb prioritású operátorokat és kiírjuk azokat,
2. Ezt az operátort betesszük a verembe.

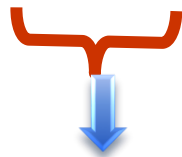


A következő szimbólum **operandus**: írjuk ki.

x: (1 + 2) * (3 - 4)

y: 1 2 +

s: (+



A következő szimbólum csukózárójel:

1. Írjuk ki a verem tetején lévő elemeket egészen a nyitózárójelig.

X: (1 + 2) * (3 - 4)

Y: 1 2 +

S: (

A következő szimbólum **csukózárójel**:

1. Írjuk ki a verem tetején lévő elemeket egészen a nyitózárójelig.
2. Vegyük ki a verem tetejéről a nyitózárójelet.

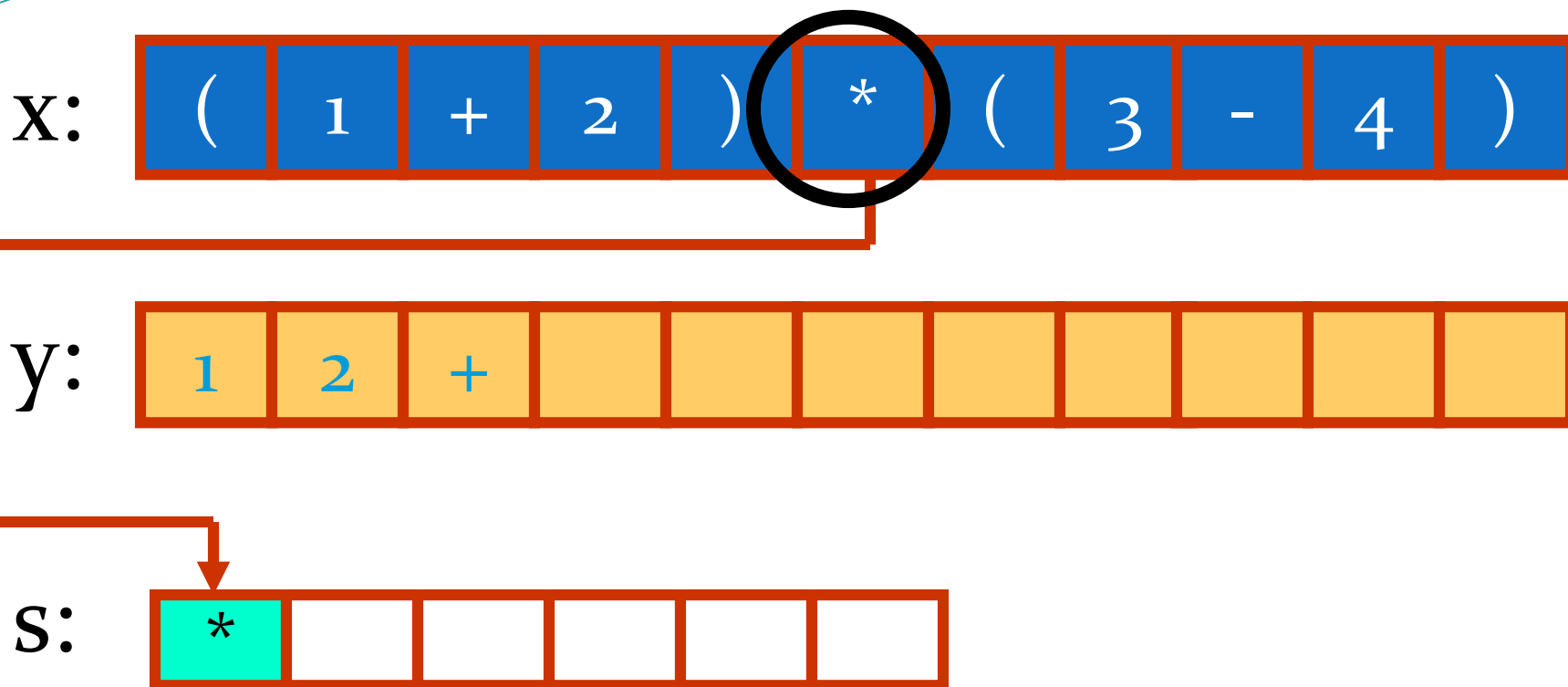
X: (1 + 2) * (3 - 4)

Y: 1 2 +

S:

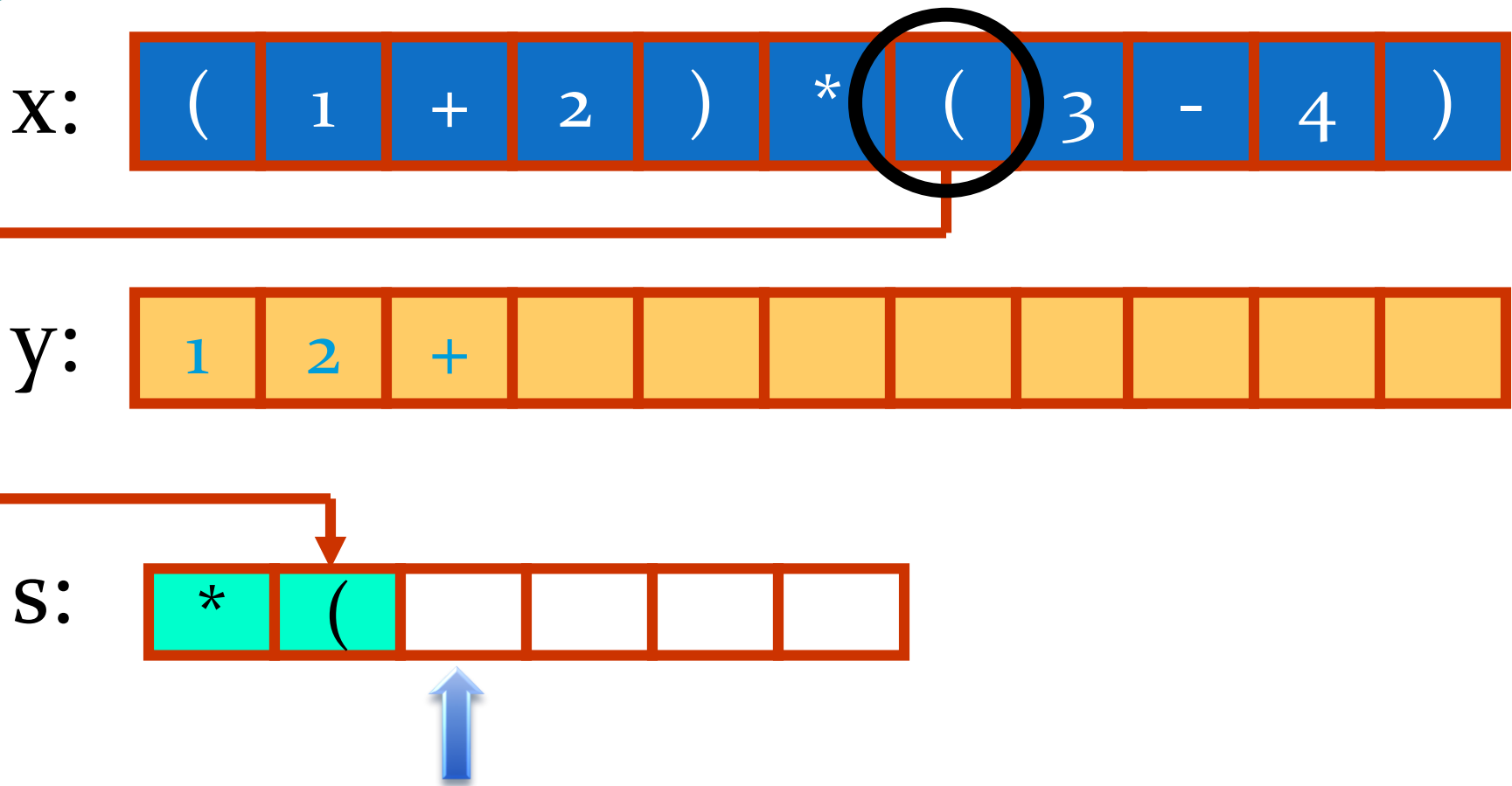
A következő szimbólum **operátor**:

1. **legfeljebb a nyitózárójelig kivesszük a veremből az operátornál nagyobb prioritású operátorokat és kiírjuk azokat,**

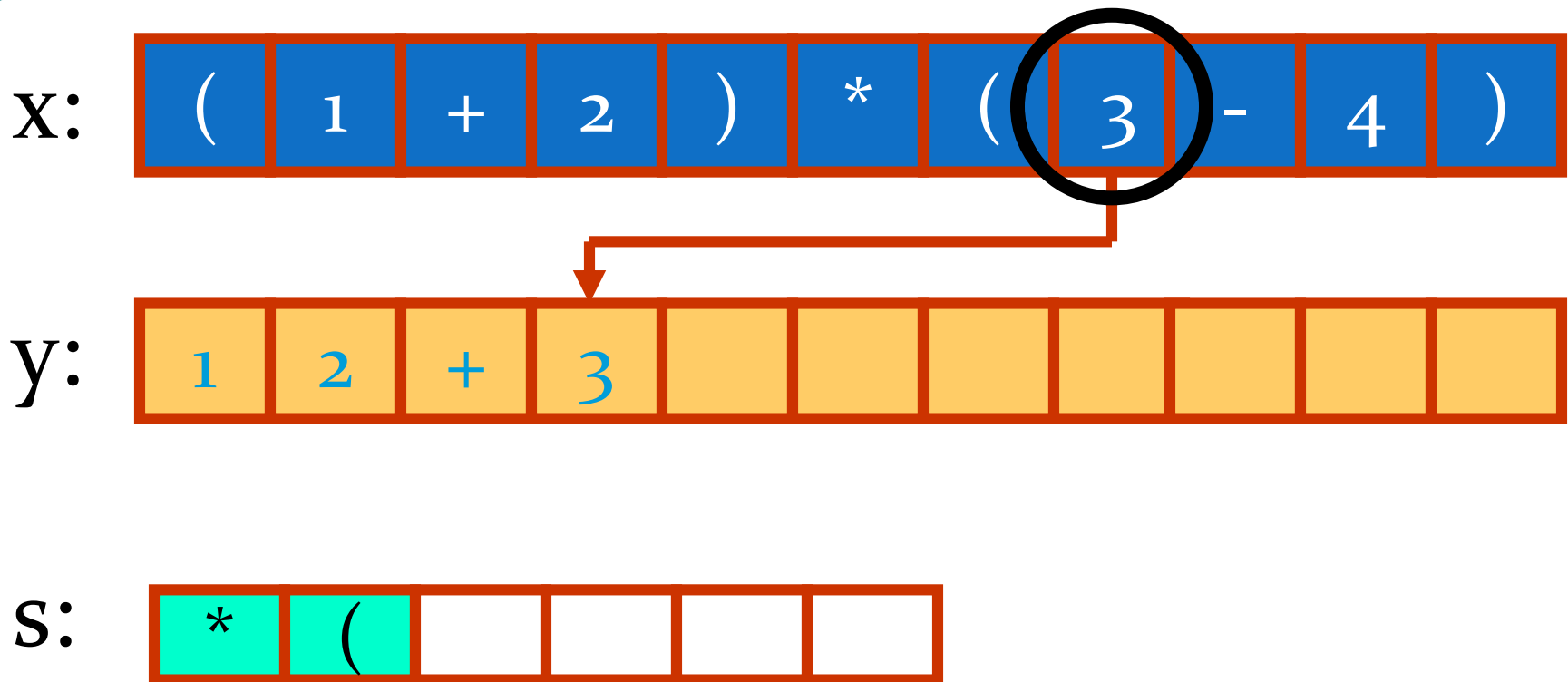


A következő szimbólum **operátor**:

1. legfeljebb a nyitózárójelig kivesszük a veremből az operátornál nagyobb prioritású operátorokat és kiírjuk azokat,
2. Ezt az operátort betesszük a verembe.



A következő szimbólum **nyitózárójel**: **tegyük a verembe.**



A következő szimbólum **operandus**: írjuk ki.

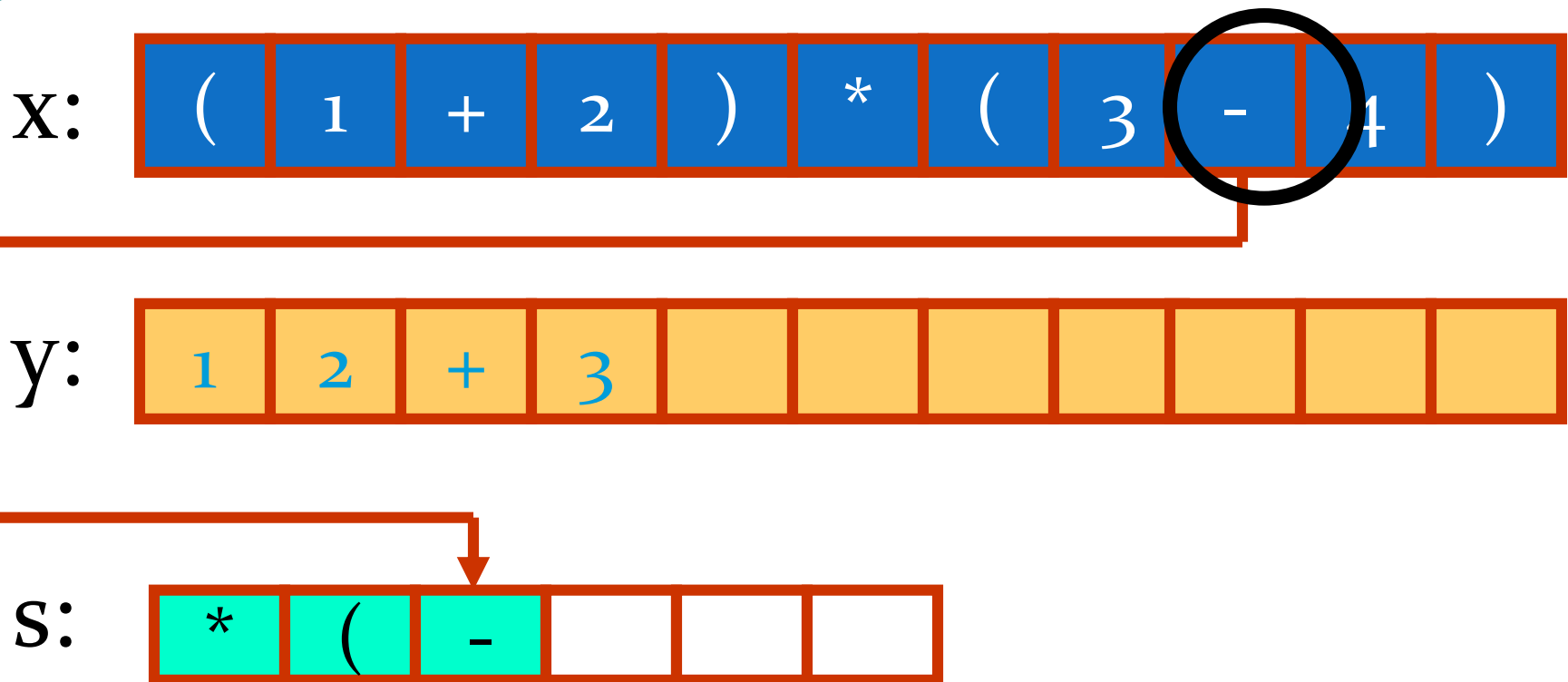
X: (1 + 2) * (3 - 4)

Y: 1 2 + 3

S: * (

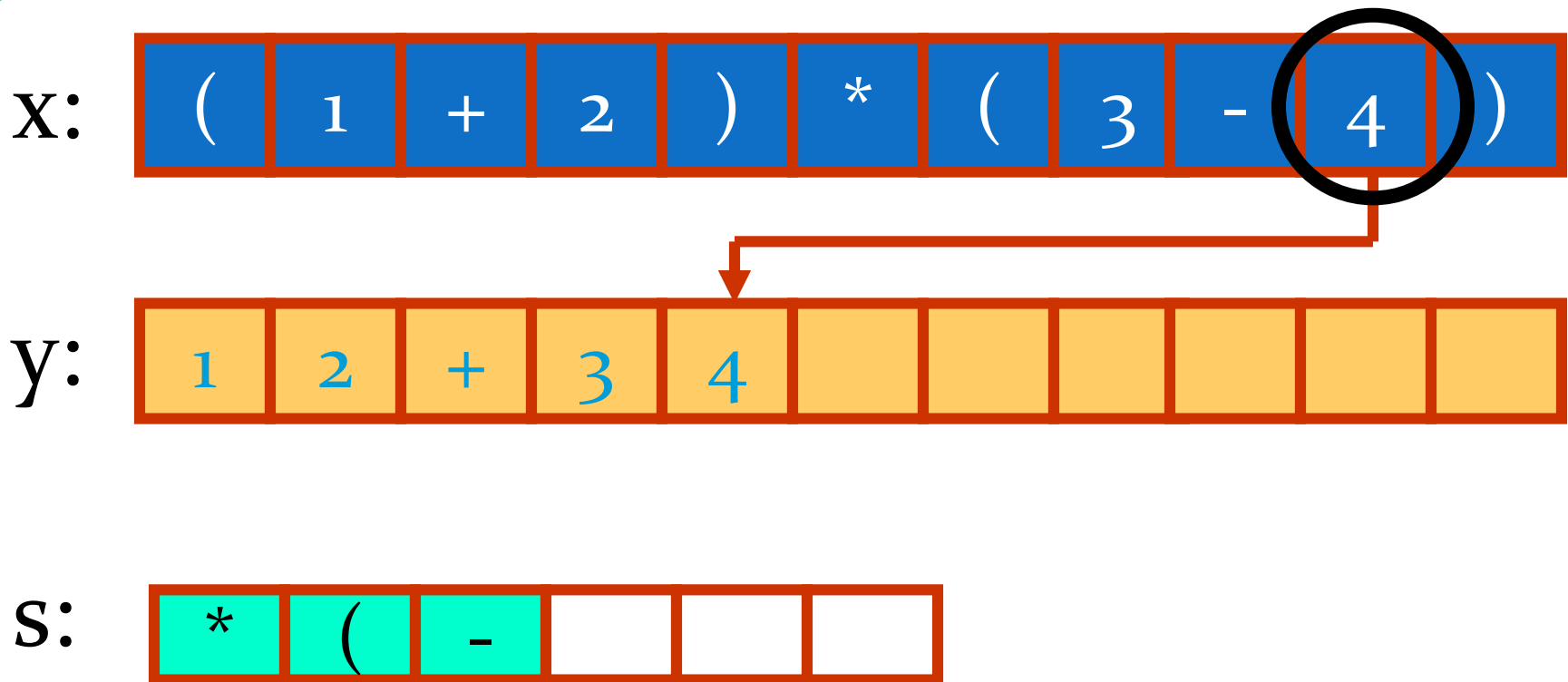
A következő szimbólum **operátor**:

1. legfeljebb a nyitózárójelig kivesszük a veremből az operátornál nagyobb prioritású operátorokat és **kiírjuk** azokat,



A következő szimbólum **operátor**:

1. legfeljebb a nyitózárójelig kivesszük a veremből az operátornál nagyobb prioritású operátorokat és kiírjuk azokat,
2. Ezt az operátort betesszük a verembe.



A következő szimbólum operandus: írjuk ki.

X: (1 + 2) * (3 - 4)

Y: 1 2 + 3 4 -

S: * (-

A következő szimbólum csukózárójel:
1. Írjuk ki a verem tetején lévő elemeket egészen a nyitózárójelig.

X: (1 + 2) * (3 - 4)

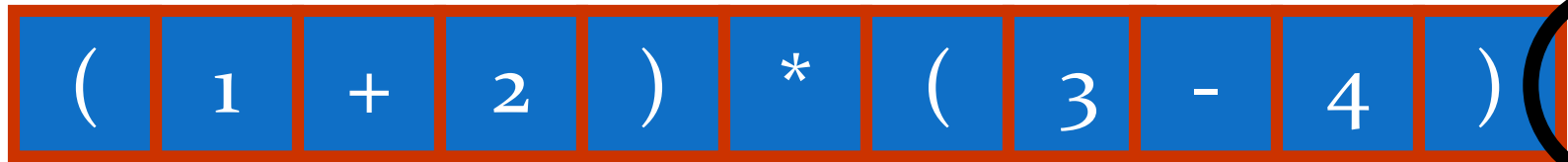
Y: 1 2 + 3 4 -

S: * (

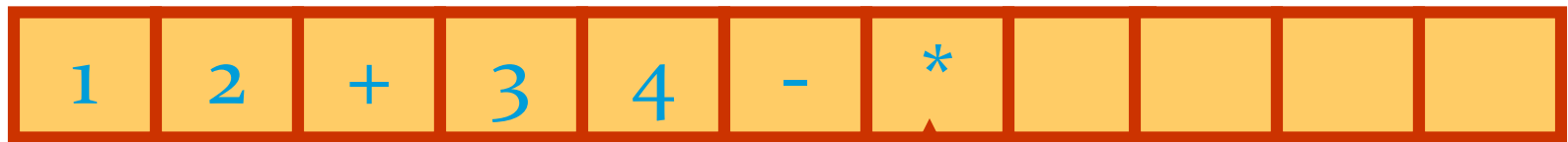
A következő szimbólum **csukózárójel**:

1. Irjuk ki a verem tetején lévő elemeket egészen a nyitózárójelig.
2. Vegyük ki a verem tetejéről a nyitózárójelet.

x:



y:



s:



Elértük a kifejezés végét:

Írjuk ki a veremben lévő összes elemet.

Kiértékelés

- $1\ 2\ +\ 3\ 4\ -\ *$
 - Kiértékelés után
 - -3

Y:

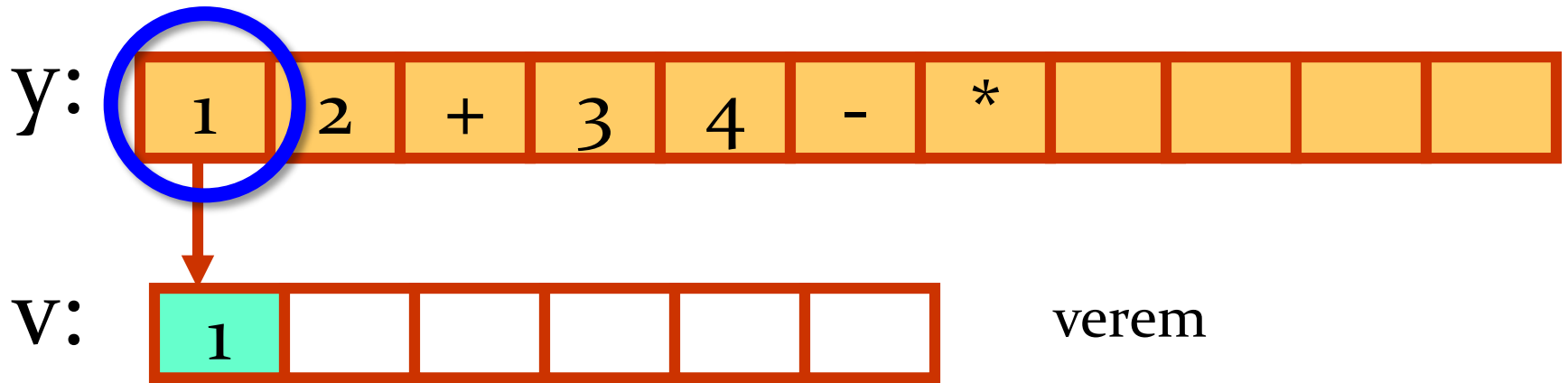
1	2	+	3	4	-	*				
---	---	---	---	---	---	---	--	--	--	--

sorozat

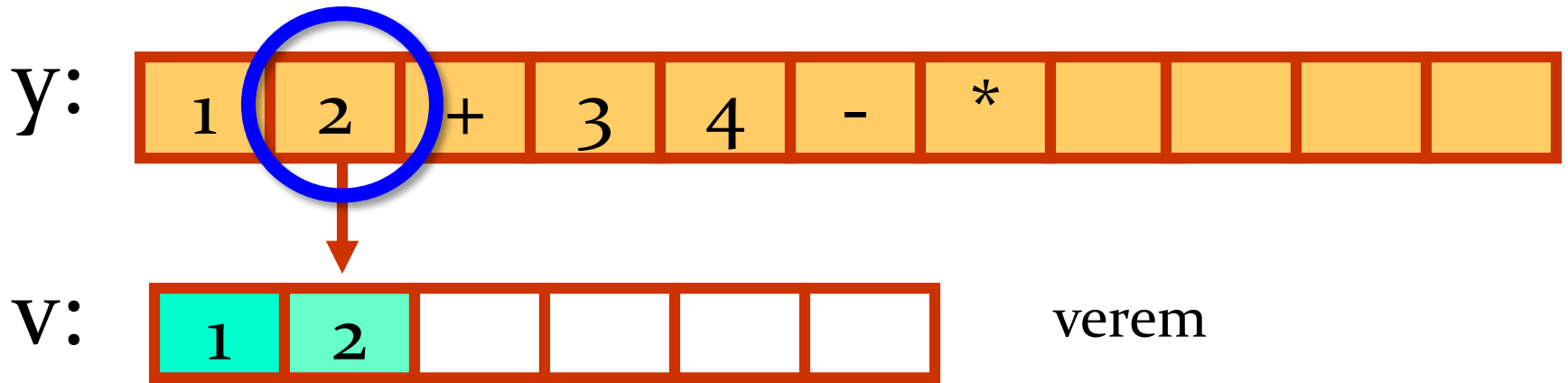
V:

--	--	--	--	--	--

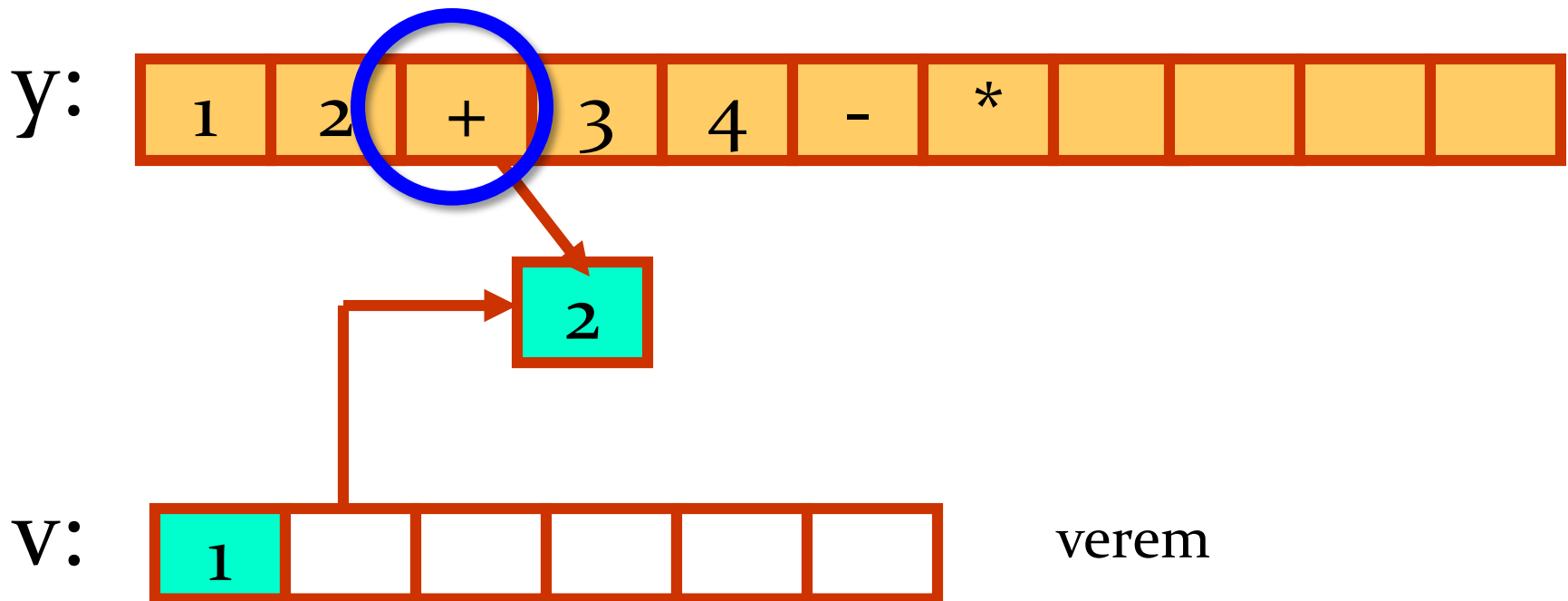
verem



A következő szimbólum **operandus**:
Tegyük a verembe.

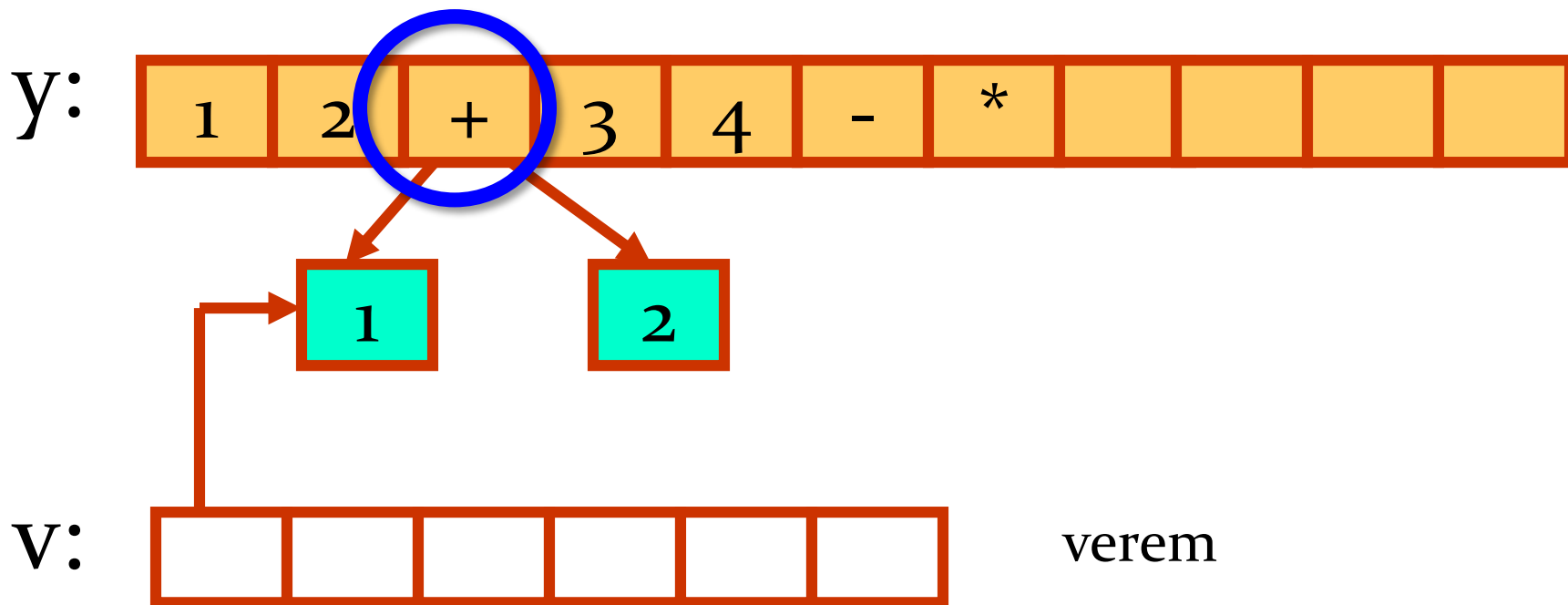


A következő szimbólum **operandus**:
Tegyük a verembe.



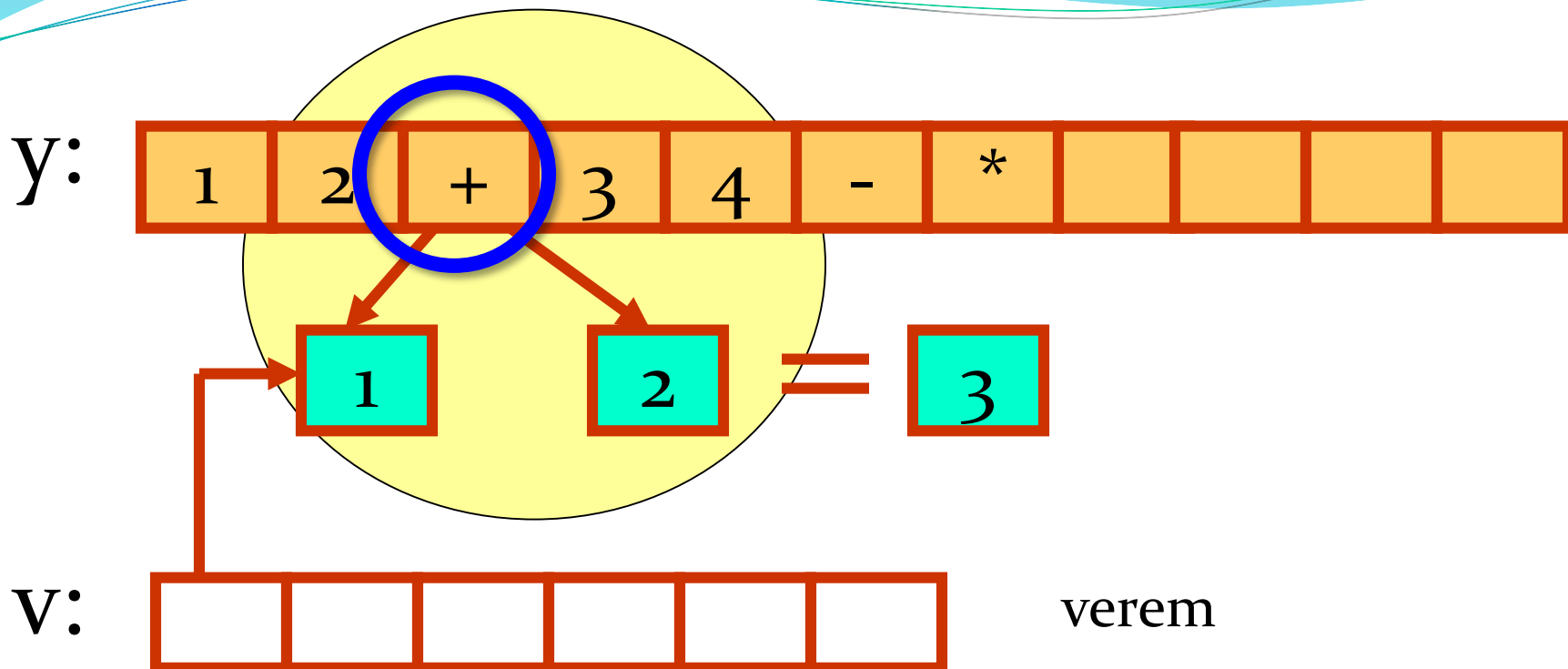
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust.



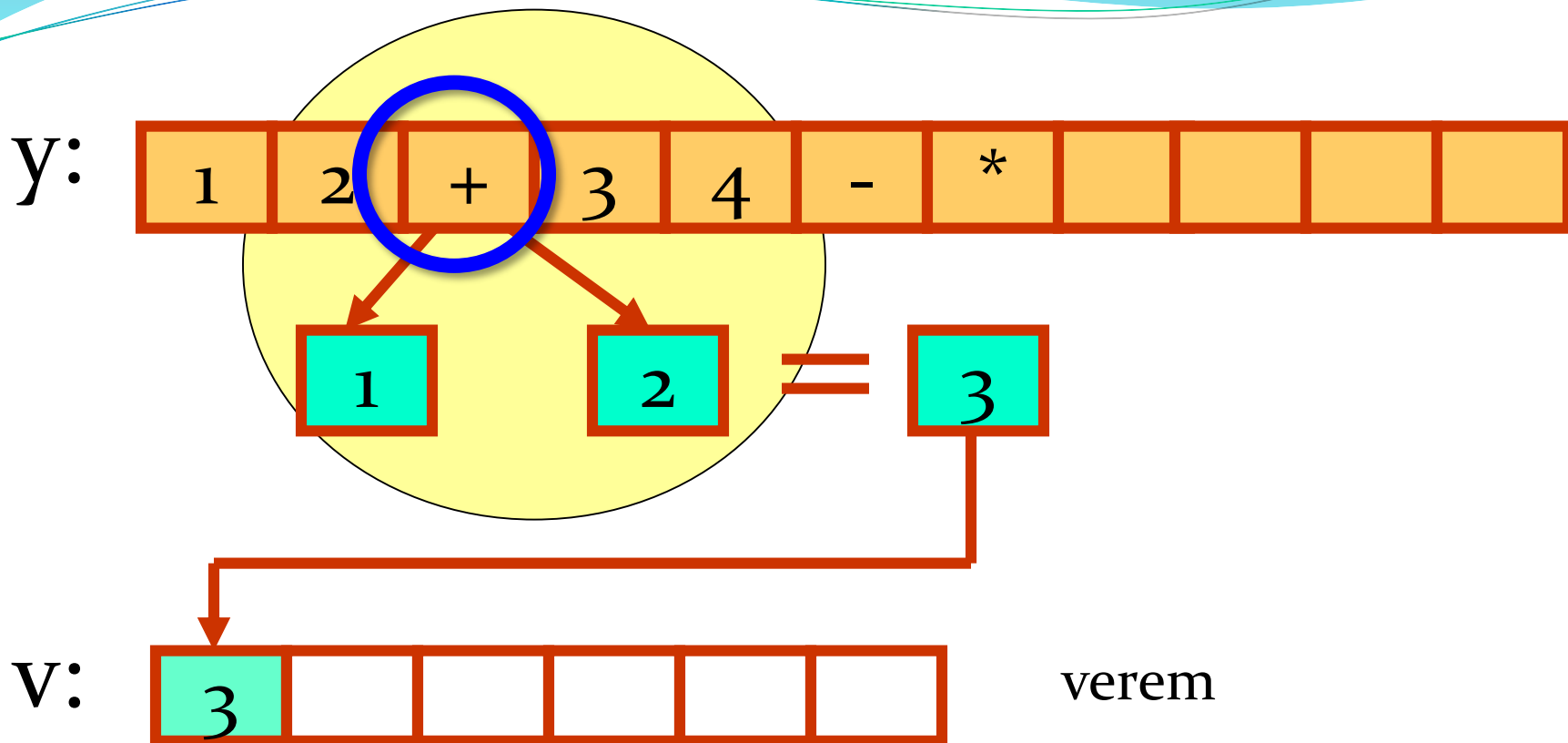
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.



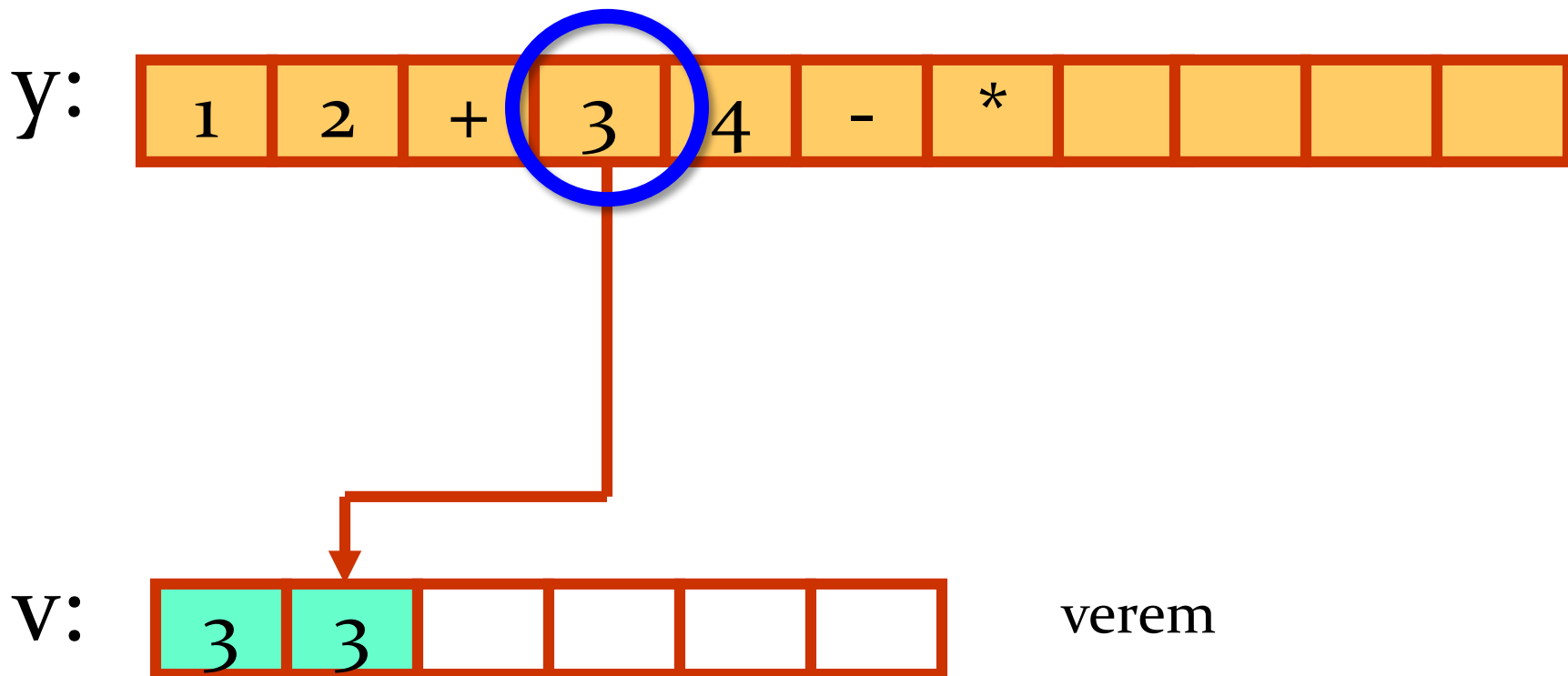
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.
3. Számítsuk ki az adott műveleti jellel a kifejezés értékét.

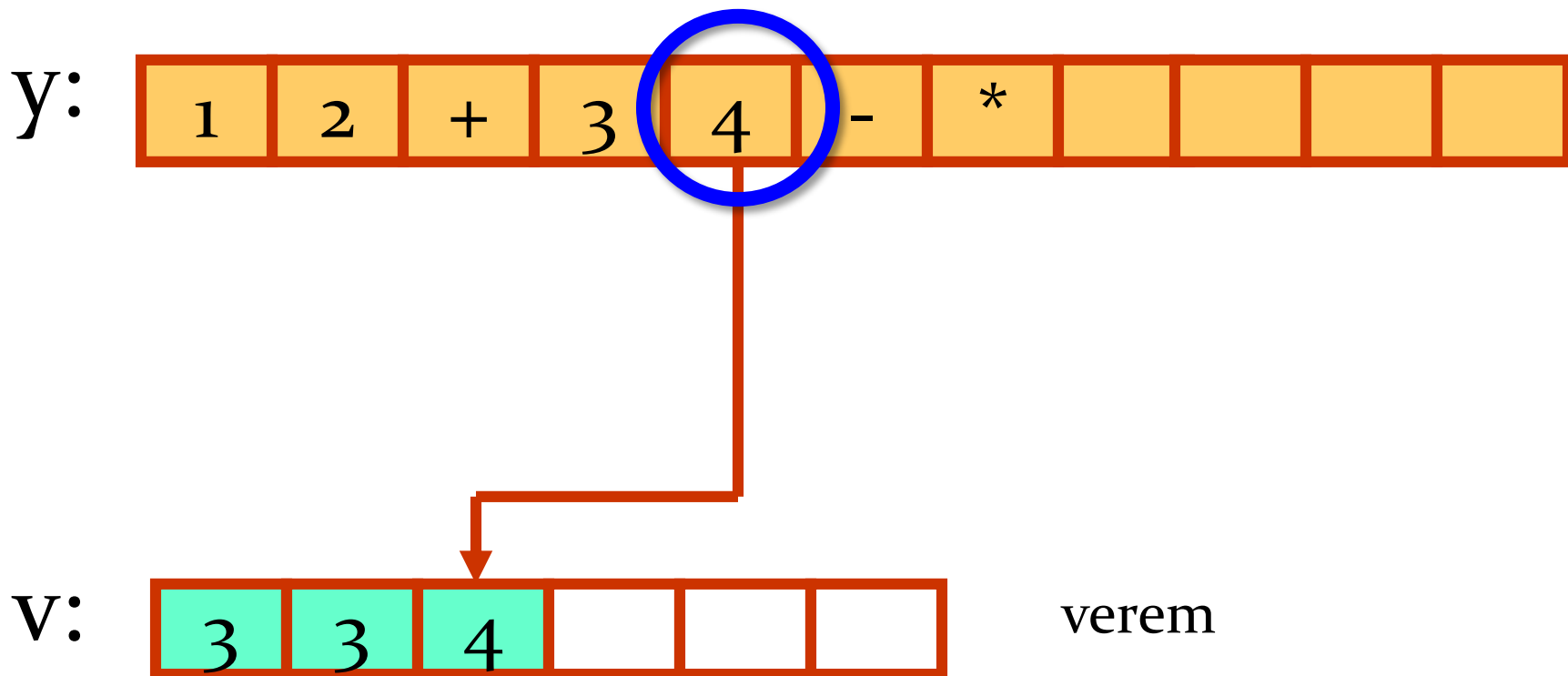


A következő szimbólum **operátor**:

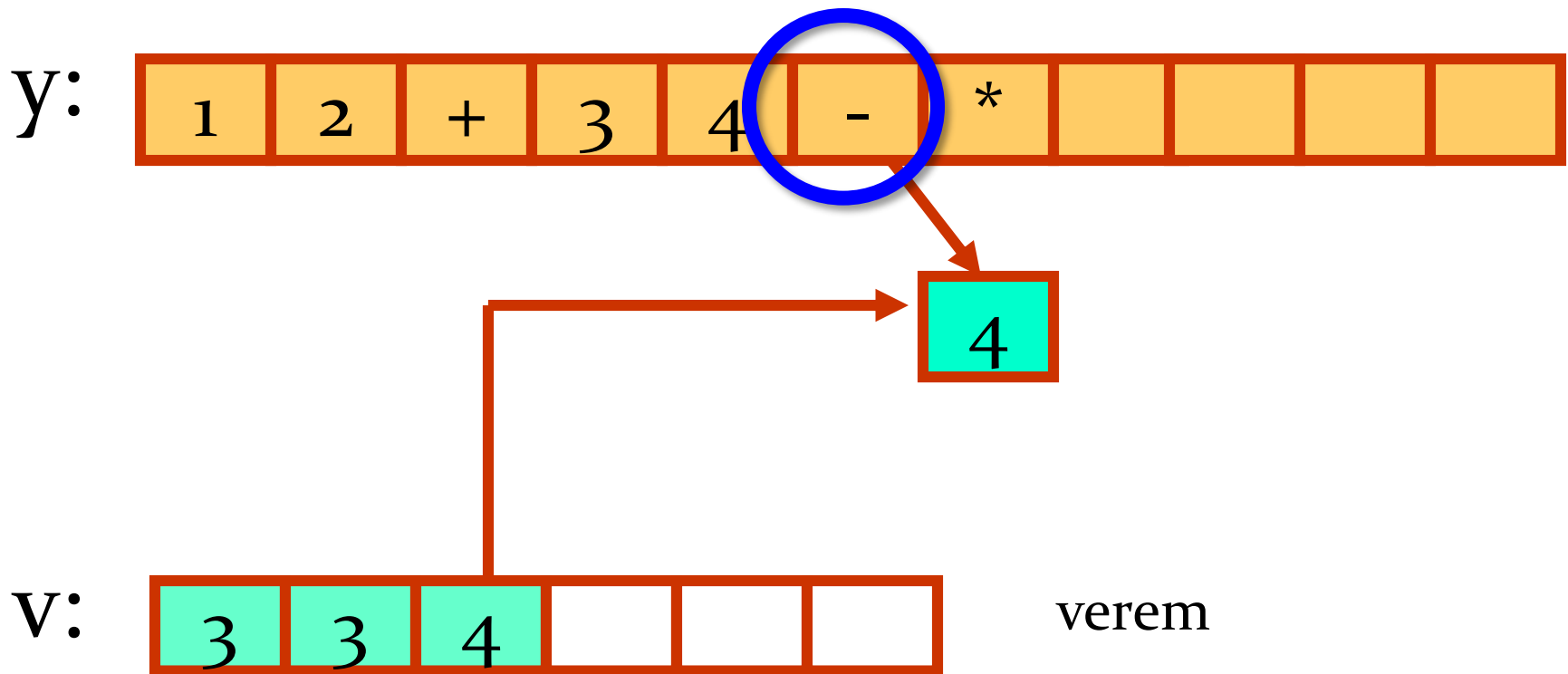
1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.
3. Számítsuk ki az adott műveleti jellel a kifejezés értékét.
4. Az így kapott eredményt tegyük a verembe.



A következő szimbólum **operandus**:
Tegyük a verembe.

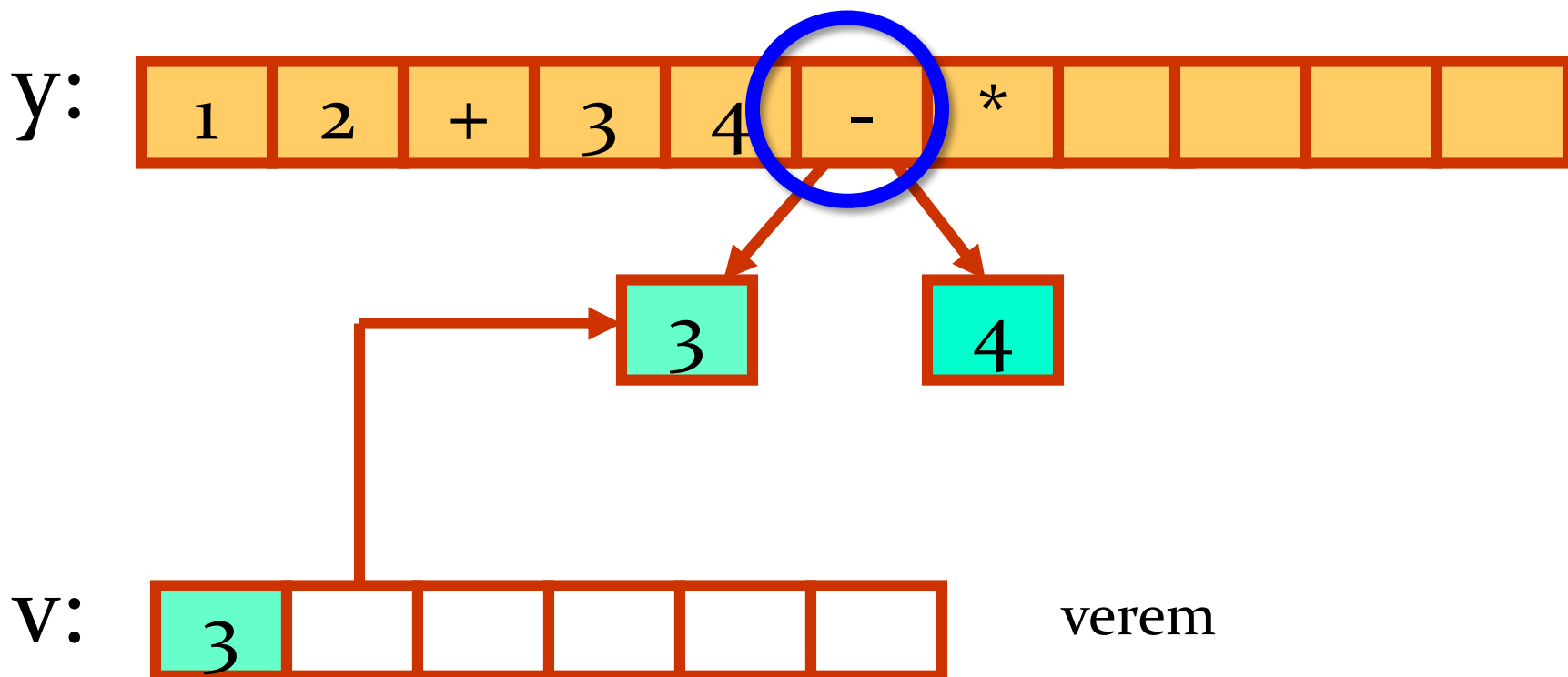


A következő szimbólum **operandus**:
Tegyük a verembe.



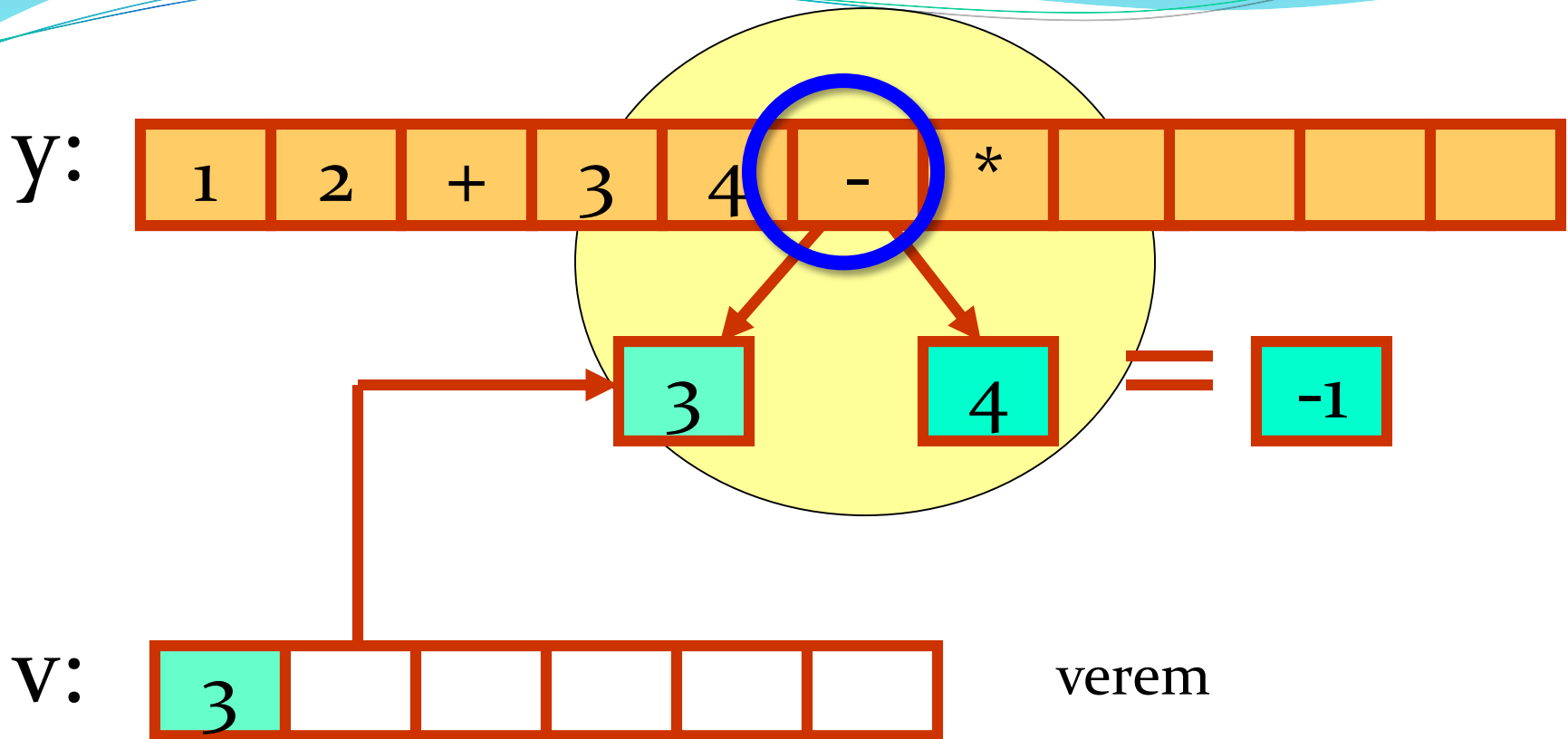
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust.



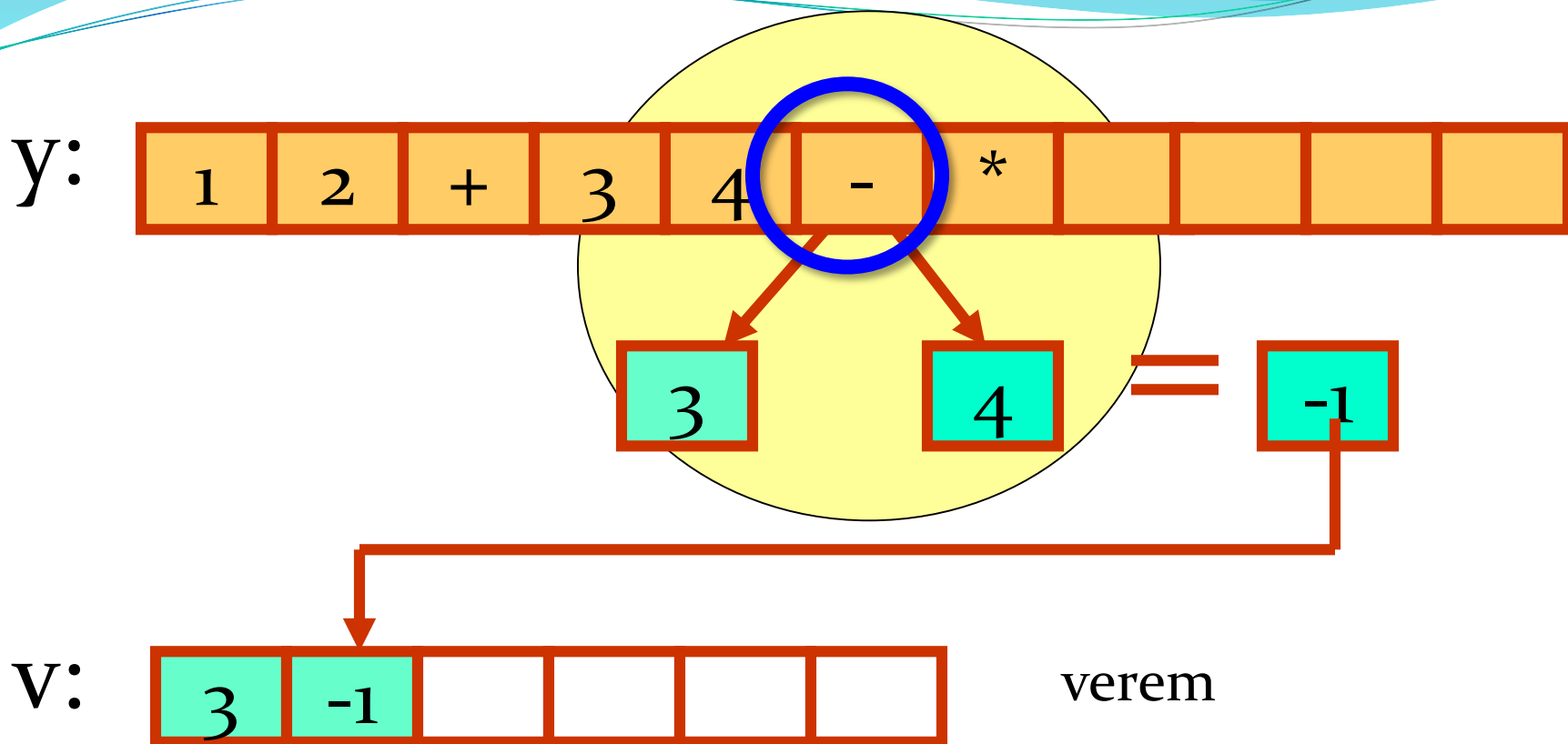
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.



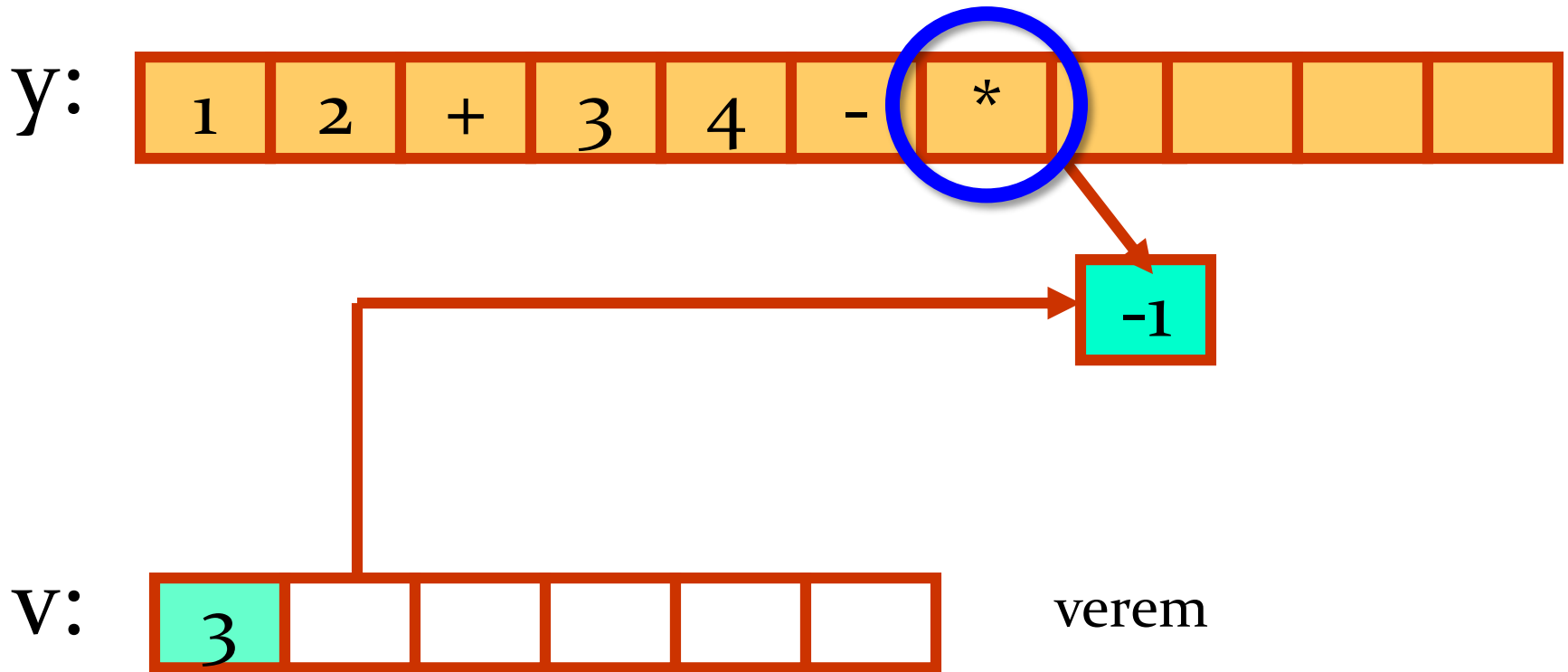
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.
3. Számítsuk ki az adott műveleti jellel a kifejezés értékét.



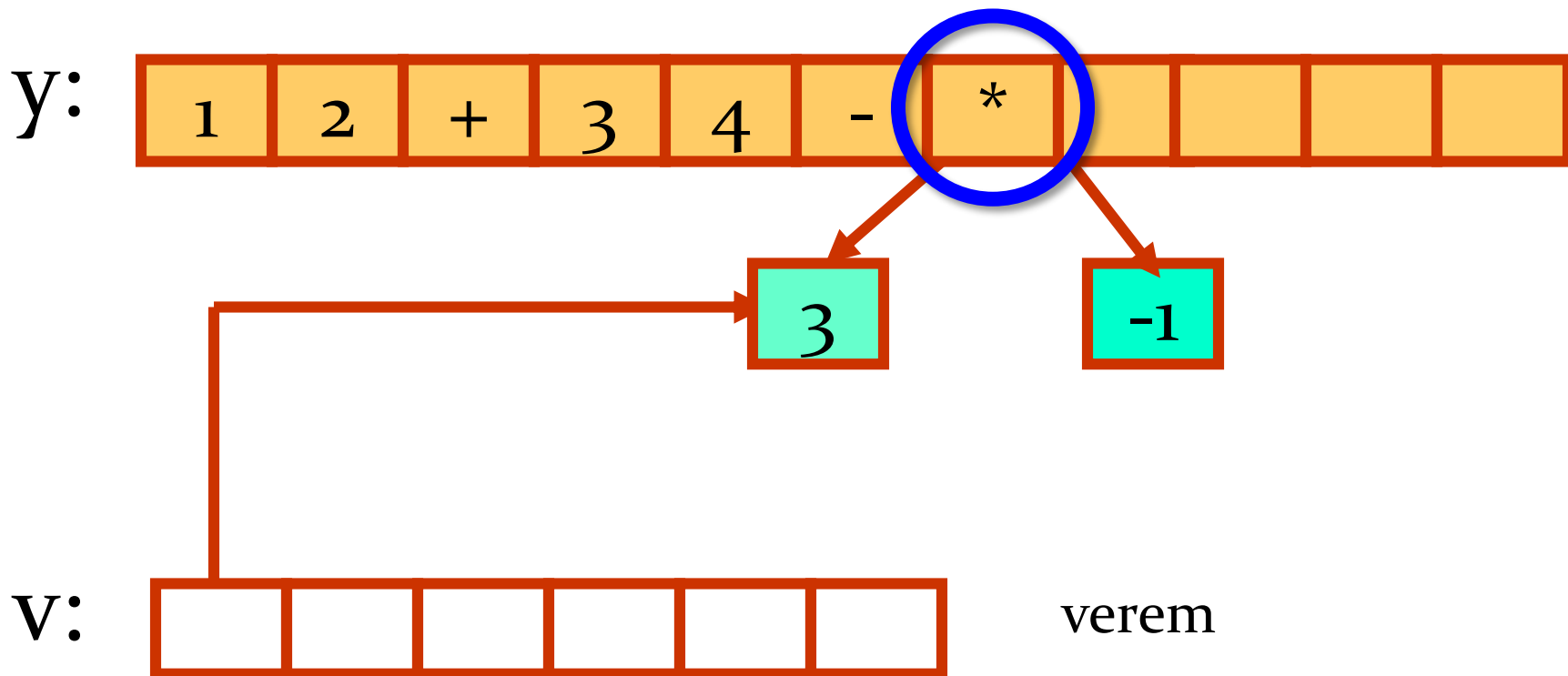
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.
3. Számítsuk ki az adott műveleti jellel a kifejezés értékét.
4. Az így kapott eredményt tegyük a verembe.



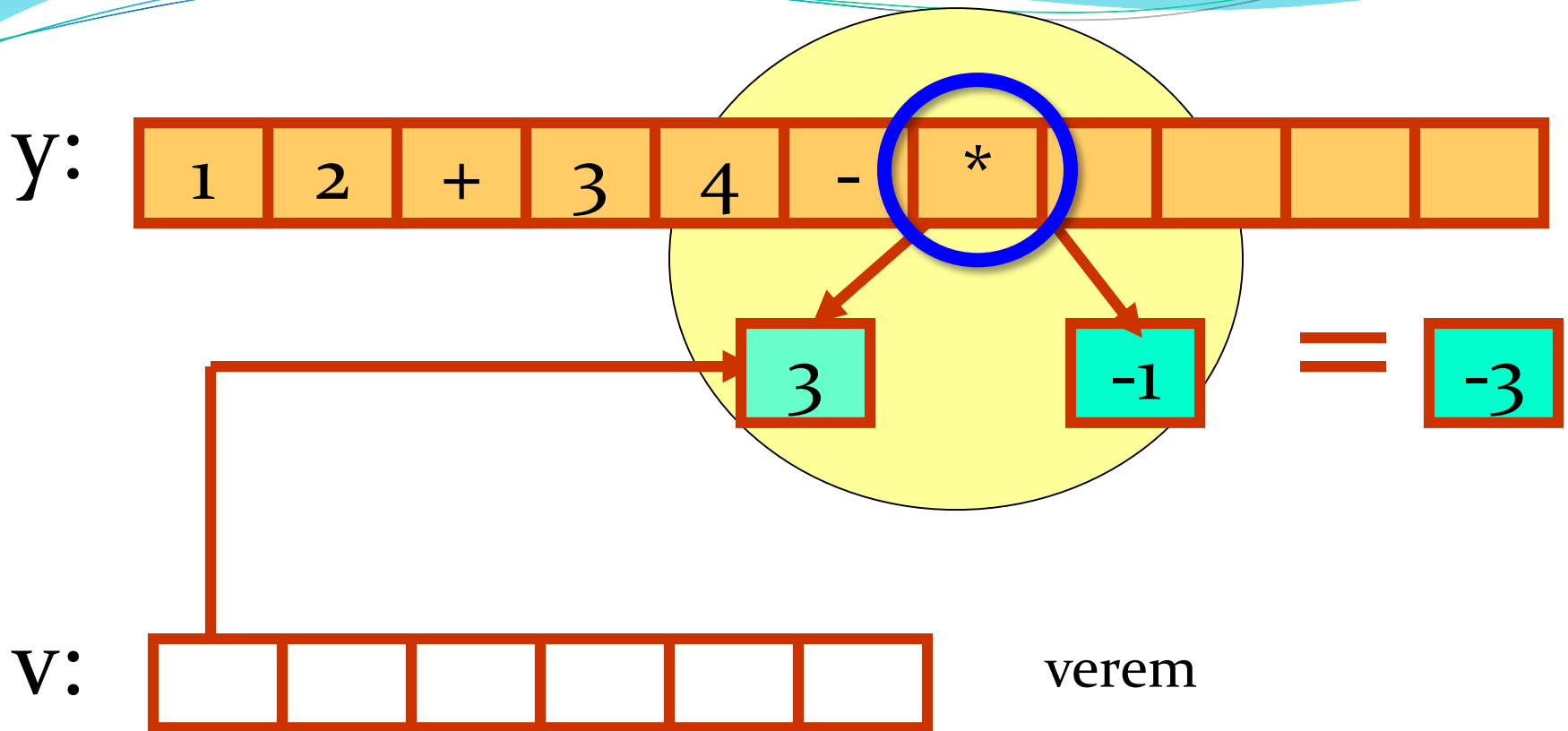
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust.



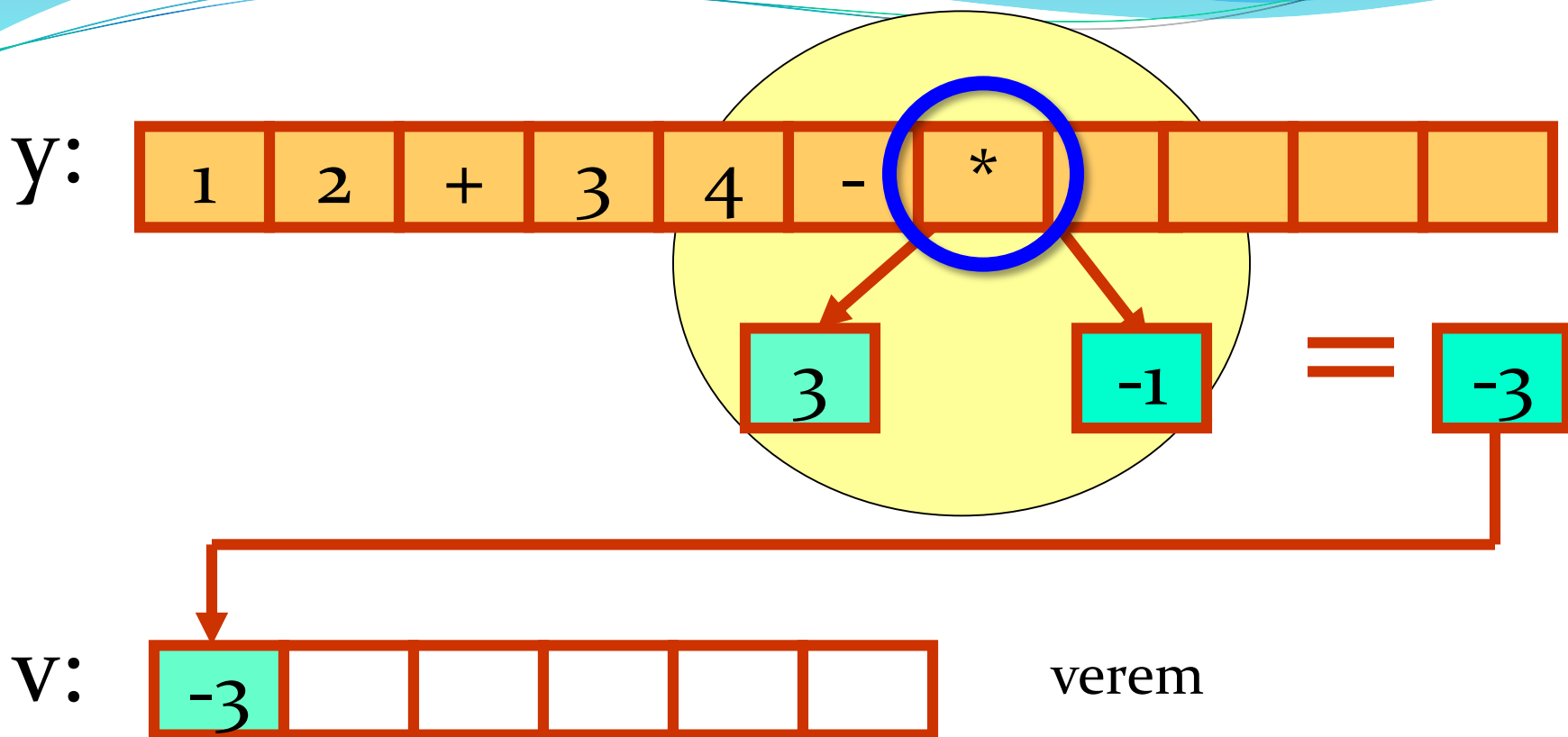
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.



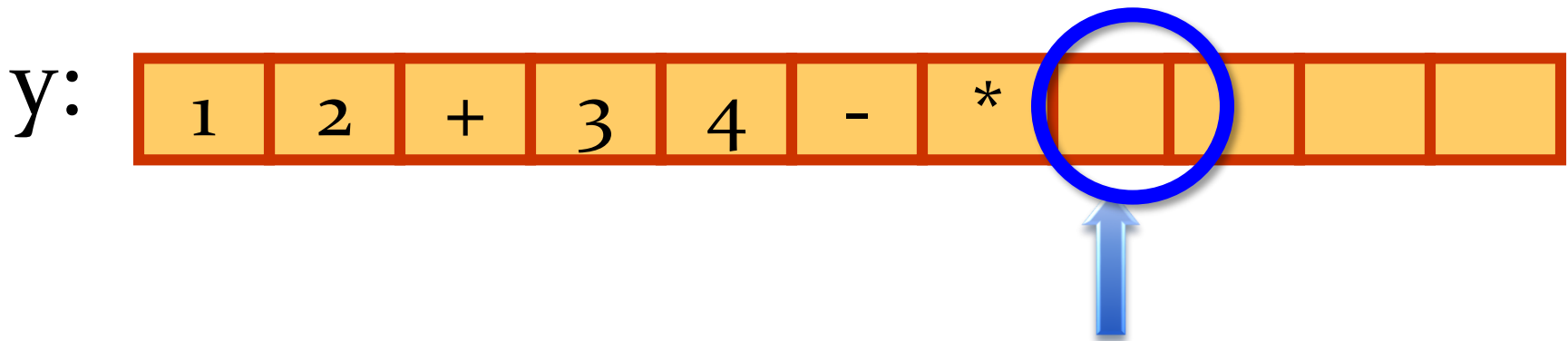
A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.
3. Számítsuk ki az adott műveleti jellel a kifejezés értékét.

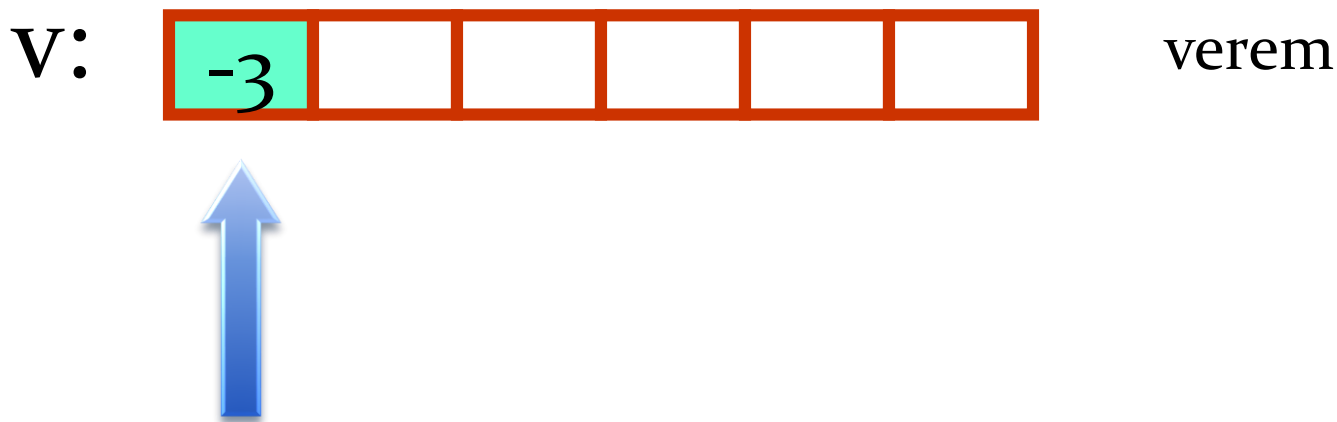


A következő szimbólum **operátor**:

1. Vegyük ki a veremből a második operandust
2. Vegyük ki a veremből az első operandust.
3. Számítsuk ki az adott műveleti jellel a kifejezés értékét.
4. Az így kapott eredményt tegyük a verembe.



Elértük a kifejezés végét



Az eredmény a verem tetején van.

Lengyel formára alakítás

- Tegyük fel, hogy az x „token”-ekből álló sor tartalmazza a szintaktikailag helyes kifejezést,
- Egy token lehet:
 - Operandus, vagy
 - Operátor (bináris), vagy
 - „(” vagy „)”
 - az y sorba hozzuk létre a postfix formájú kifejezést,
- közben felhasználva az s vermet, mely operátorokat és nyitózárójelet tartalmazhat

y.empty; s.empty

$\neg x.isempty$

$e \leftarrow x.out$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg s.isempty$
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.isempty$

y.in(s.pop)

y.empty; s.empty

$\neg x.isempty$

$e \leftarrow x.out$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg s.isempty$
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.isempty$

y.in(s.pop)

y.empty; s.empty

$\neg x.\text{isempty}$

$e \leftarrow x.\text{out}$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg s.\text{isempty}$
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.\text{isempty}$

y.in(s.pop)

y.empty; s.empty

$\neg x.\text{isempty}$

$e \leftarrow x.\text{out}$

I / e=operandus? / N			
I / y.in(e)	I / e='('	I / e=')'	I / e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg$ s.isempty
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.\text{isempty}$

y.in(s.pop)

y.empty; s.empty

$\neg x.\text{isempty}$

$e \leftarrow x.\text{out}$

I / e=operandus?		N	
I / e='('		I / e=')'	I / e=operator
y.in(e)	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg$ s.isempty
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.\text{isempty}$

y.in(s.pop)

y.empty; s.empty

$\neg x.\text{isempty}$

$e \leftarrow x.\text{out}$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg$ s.isempty
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.\text{isempty}$

y.in(s.pop)

y.empty; s.empty

$\neg x.\text{isempty}$

$e \leftarrow x.\text{out}$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg$ s.isempty
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.\text{isempty}$

y.in(s.pop)

y.empty; s.empty				
¬x.isempty				
e←x.out				
I / e=operandus? / N				
y.in(e)	I / e='('	I / e=')'	I / e=operator	
	s.push(e)	s.top ≠ '('	s.top ≠ '(' ∧ prec(s.top) ≥ prec(e) ∧ ¬s.isempty	
		y.in(s.pop)	y.in(s.pop)	
		s.pop	s.push(e)	
¬s.isempty				
y.in(s.pop)				

y.empty; s.empty

$\neg x.isempty$

$e \leftarrow x.out$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg s.isempty$
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.isempty$

y.in(s.pop)

y.empty; s.empty

$\neg x.\text{isempty}$

$e \leftarrow x.\text{out}$

I \ e=operandus?		N	
y.in(e)	I \ e='('	I \ e=')'	I \ e=operator
	s.push(e)	s.top $\neq$ '('	s.top $\neq$ '(' $\wedge$ prec(s.top) $\geq$ prec(e) $\wedge$ $\neg$ s.isempty
		y.in(s.pop)	y.in(s.pop)
		s.pop	s.push(e)

$\neg s.\text{isempty}$

y.in(s.pop)

Lengyel forma kiértékelése

- Tegyük fel, hogy az y sor tartalmazza a postfix alakban lévő kifejezést,
- Értékeljük ki, felhasználva a v vermet, mely operandusokat tartalmazhat
- Az eredményt tároljuk a z változóban

y.empty; z←o

$\neg x.\text{isempty}$

e←y.out

I

e=operandus?

N

v.push(e)

op1←v.pop

op2←v.pop

r←Művelet végrehajtása

v.push(r)

z←v.pop

y.empty; z←o

$\neg x.\text{isempty}$

e←y.out

I

e=operandus?

N

v.push(e)

op1←v.pop

op2←v.pop

r←Művelet végrehajtása

v.push(r)

z←v.pop

y.empty; z←o

$\neg x.\text{isempty}$

$e \leftarrow y.\text{out}$

I

e=operandus?

N

v.push(e)

op1←v.pop

op2←v.pop

r←Művelet végrehajtása

v.push(r)

z←v.pop

y.empty; z ← o

¬x.isempty

e ← y.out

I

e = operandus?

N

v.push(e)

op1 ← v.pop

op2 ← v.pop

r ← Művelet végrehajtása

v.push(r)

z ← v.pop

y.empty; z←o

$\neg x.\text{isempty}$

e←y.out

I / e=operandus?

N /

v.push(e)

op1←v.pop

op2←v.pop

r←Művelet végrehajtása

v.push(r)

z←v.pop

y.empty; z←o

$\neg x.\text{isempty}$

e←y.out

I / e=operandus?

N /

v.push(e)

op1←v.pop
op2←v.pop
r←Művelet végrehajtása
v.push(r)

z←v.pop

y.empty; z←o

$\neg x.\text{isempty}$

e←y.out

I / e=operandus?

N /

v.push(e)

op1←v.pop
op2←v.pop
r←Művelet végrehajtása
v.push(r)

z←v.pop

y.empty; z←o

$\neg x.\text{isempty}$

e←y.out

I / e=operandus?

N /

v.push(e)

op1←v.pop
op2←v.pop
r←Művelet végrehajtása
v.push(r)

z←v.pop