$v : Stack(n)$ ← max ilyen hosszú

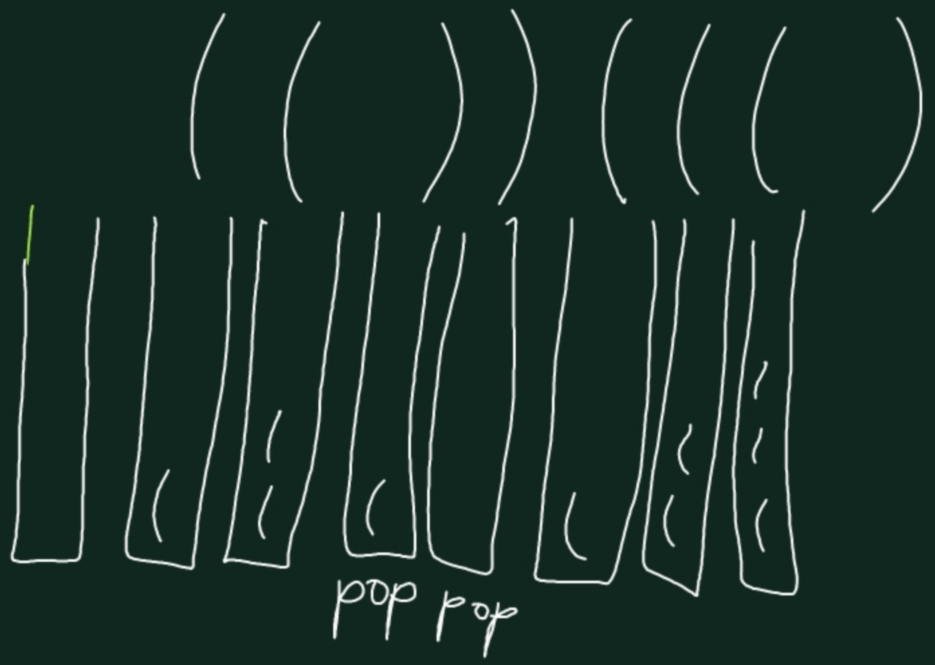
$v.isEmpty() : \mathbb{B}$ ← üres-e?

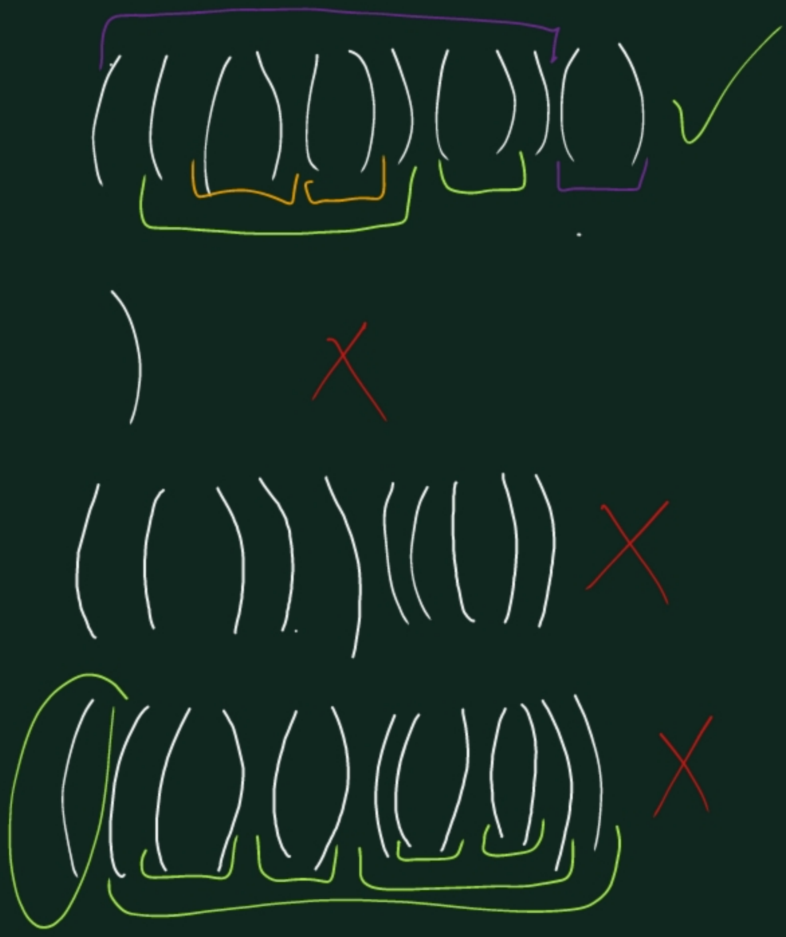
$v.push(x)$

$v.pop() : T$

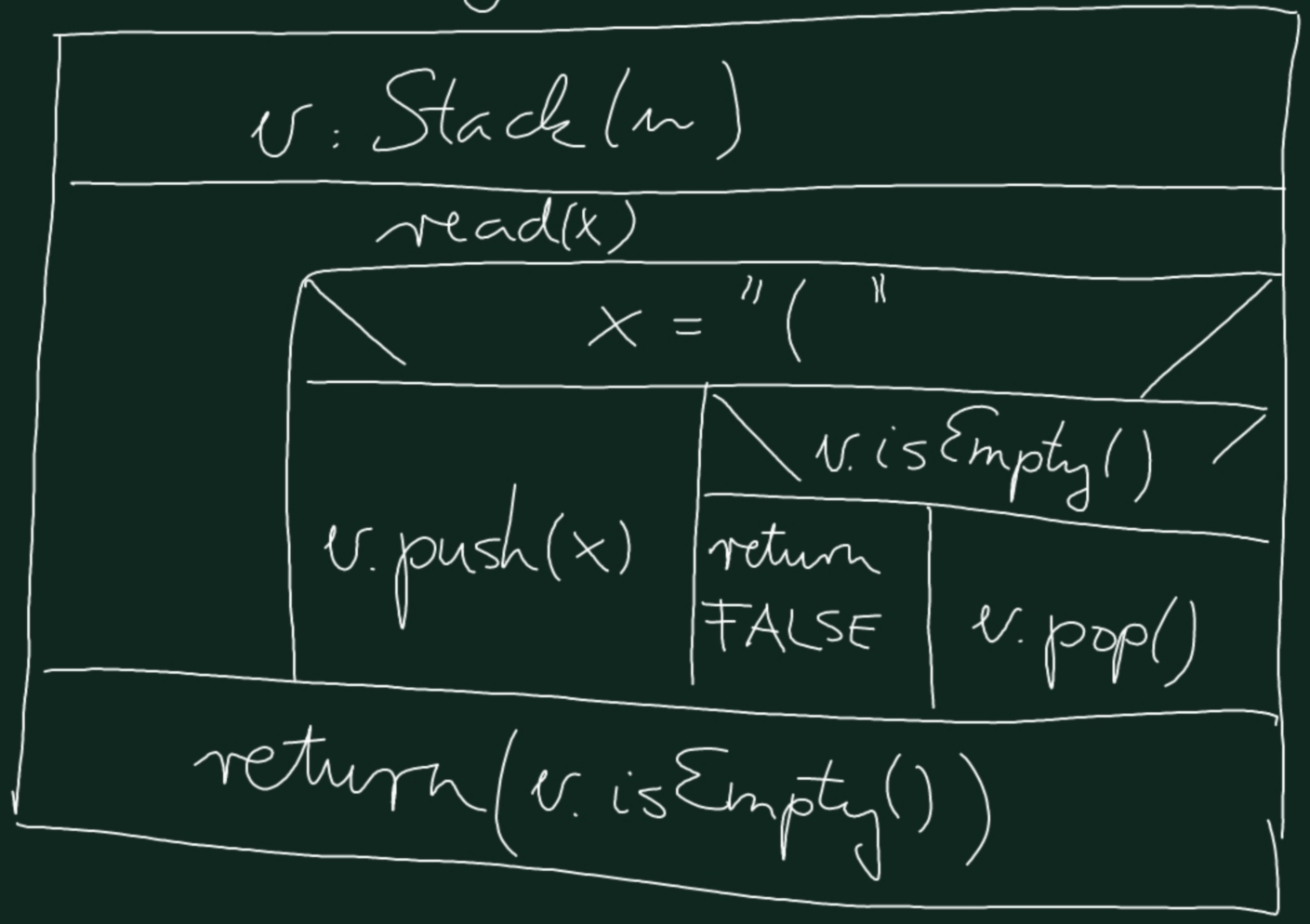
$v.top() : T$

Adott egy zárójelekből álló, legfeljebb n hosszú karaktersorozat a bemeneten. Olvassuk be, és döntsük el róla, hogy helyes zárójelezést határoz-e meg! (Vagyis, hogy párba állíthatók-e a zárójelek úgy, hogy minden nyitó zárójelnek van egy olyan csukó zárójel a párja, amely később következik a sorozatban.)





Zárójel(n : ℕ) : 𝔹

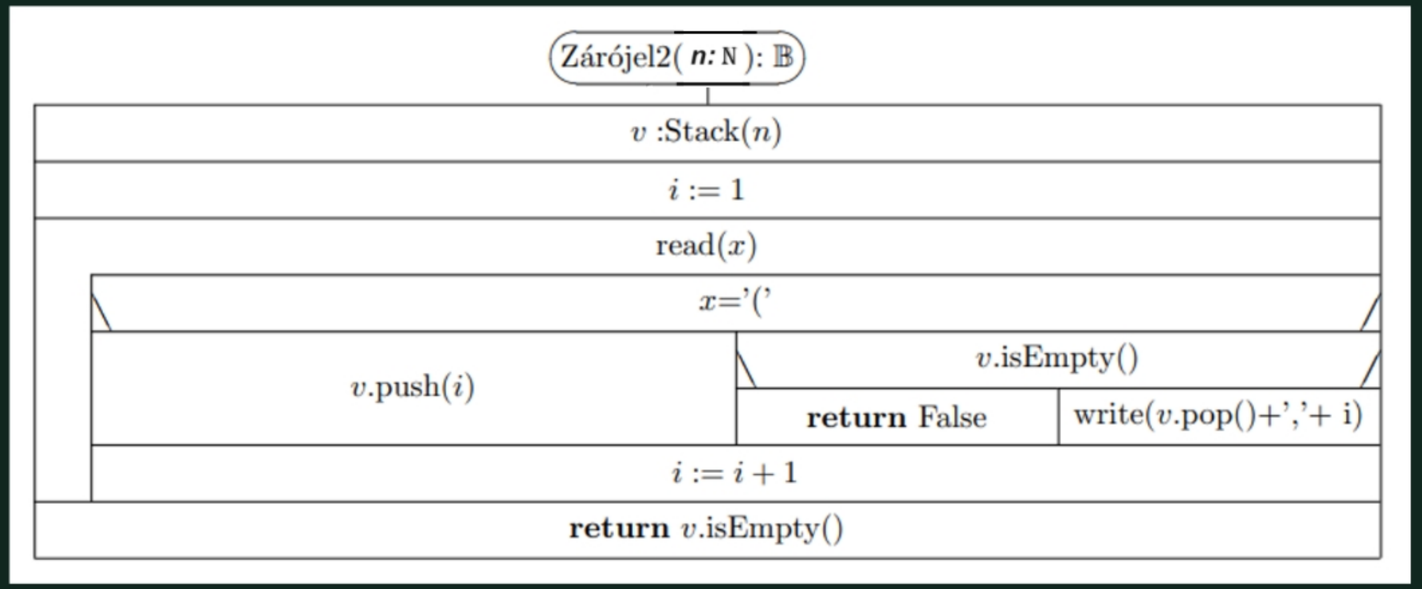


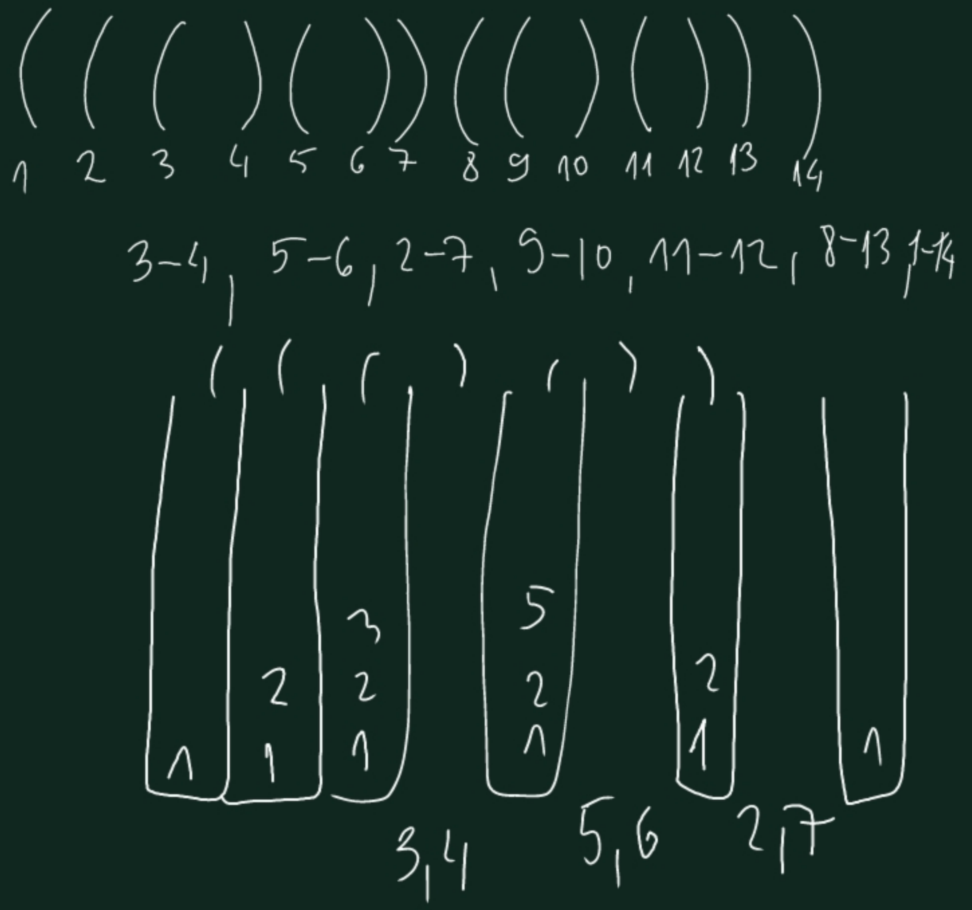
read(x)

X-be beolvassa a következő "egységet"

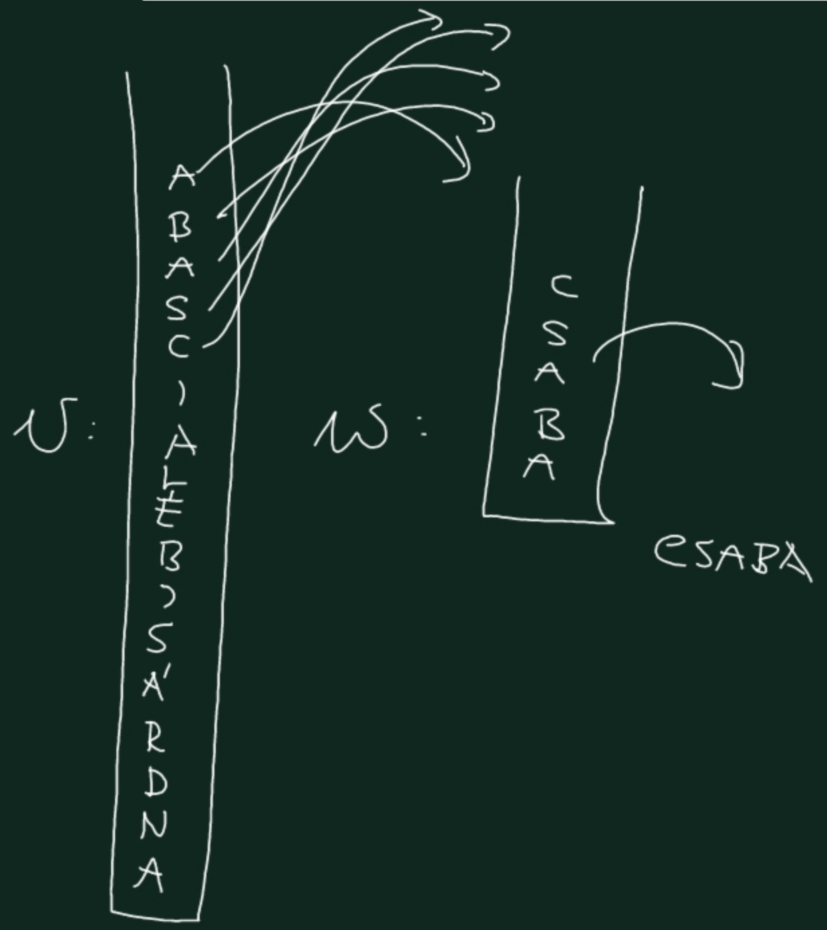
- ha sikeres a beolvasás → IGAZ
- ha nem → HAMIS

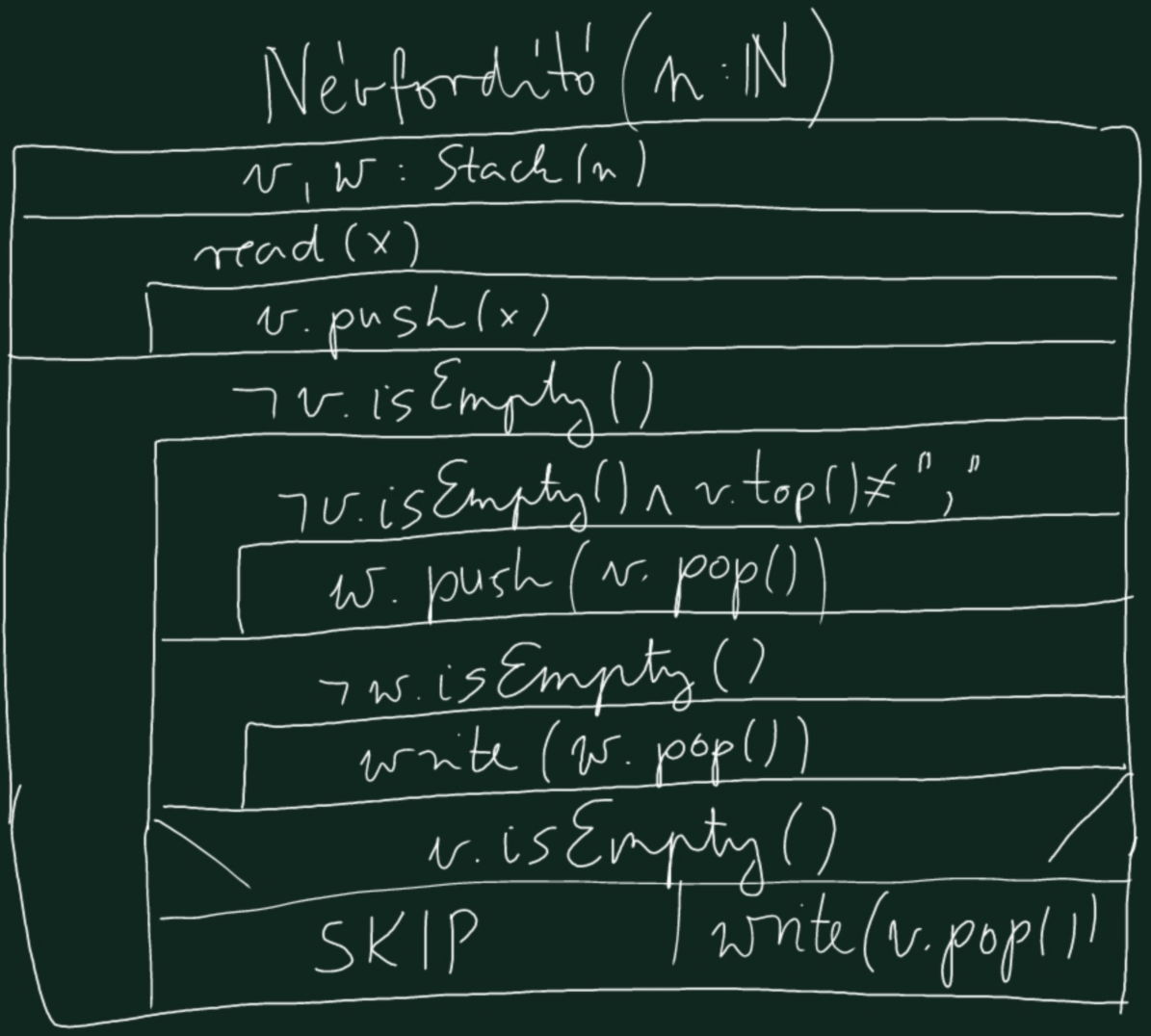
Adott egy zárójelekből álló, legfeljebb n hosszú karaktersorozat a bemeneten. Olvassuk be, és döntsük el róla, hogy helyes zárójelezést határoz-e meg! Írassuk ki az összetartozó zárójelpárok ~~indexeit!~~ sorszámait





A bemenetről karakterenként beolvasunk egy (legfeljebb n karakterből álló) névsort, ahol a nevek egymástól vesszővel vannak elválasztva. Írjuk ki a neveket fordított sorrendben! Például: András,Béla,Csaba → Csaba,Béla,András





abc * dde * ghi → cba * edd * ihg

→ abccba * ddeedd * ghiihg

→ abccbaabc * Aleeddddde *

* ....

Tükrözött-e a szöveg? Adott egy legfeljebb n hosszú, betűkből és '#' szimbólumokból álló sorozat. A sorozatot tükrözöttnek nevezzük, ha felbontható olyan páratlan hosszú, palindrom karaktersorozatokból álló részekre, amelyeknek középső karaktere az egyetlen bennük szereplő '#'. Döntsük el a bemenetről olvasott szövegről, hogy tükrözött-e!

Példák:

- Tükrözött: #, ####, abc#cba, ##a#aabc#cba
- Nem tükrözött: abc, abc#cb, abc#cbaa#aa, ab#bac##c

← Szorgalmi

Hf.

Lengyelforma

aritmetikai kifejezések
postfix alakja

$a * b + 2 * c - d / e$

infix → postfix

$a + b \longrightarrow a\,b\,+$

$a * b \longrightarrow a\,b\,*$

$a + b * c \longrightarrow a\,b\,c\,*\,+$

$a * b + c * d \longrightarrow a\,b\,*\,c\,d\,*\,+$

- konstans
- változó
- kifejezés postfix alakban

$(a+b)*c \longrightarrow ab+c*$

$a+b-c+d \longrightarrow ab+c-d+$

balról jobbra
hajtjuk végre

$a*b/c*d \longrightarrow ab*c/d*$

$a/b*c \longrightarrow ab/c*$

$a/b*c = \frac{a}{b}*c$

$\neq \frac{a}{b*c}$

a^b^c ⟶ abc^^

jobb-bal operátor

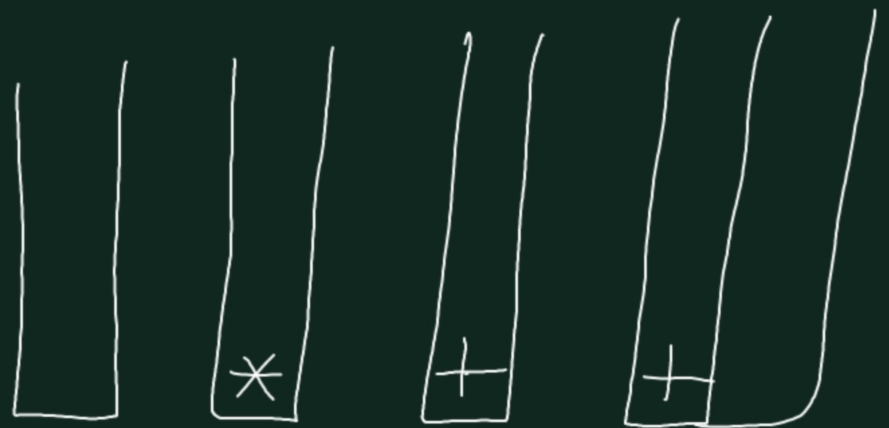
2^3^4 → $2^{3^4} = 2^{81}$

~~$(2^3)^4 = 2^{12}$~~

- Nincsenek benne zárójelek, a kiértékelés mégis egyértelmű, és könnyen elvégezhető,
- Operandusok sorrendje nem változik, az infix kifejezéshez képest,
- Operátorok sorrendje: az elvégzésük sorrendjében szerepelnek,
- Minden operátort közvetlen megelőznek az operandusai. Az operandus lehet változó, konstans, de lehet postfix kifejezés is.

infix kifejezés	lengyel forma (postfix alak)	Megjegyzés
a+b	ab+	müveleti jel az operandusai mögött áll
a+b*c	abc*+	műveletek rangsorának hatása: (prec(*) > prec(+)
a*b+c	ab*c+	műveletek rangsorának hatása: (prec(*) > prec(+)
a*(b+c)	abc+*	zárójelezés felülbírálhatja a műveletek rangsorát
a/b*c	ab/c*	azonos rangú műveletek általában balról jobbra sorrendben végzendők el
a^b^c	abc^^	a fenti szabály alól akad néhány kivétel, például az egymást követő hatványozás sorrendje jobbról balra értendő

a * b + c



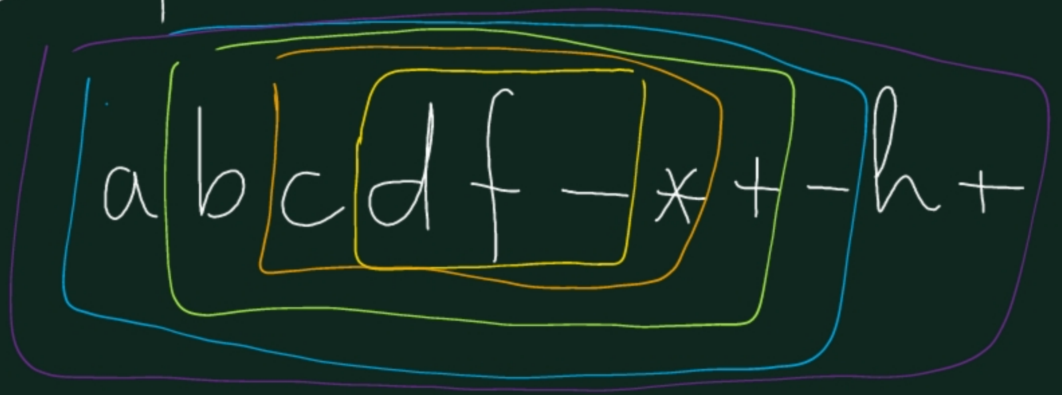
a b * c +

$a - (b + c * (d - f))) + h$

prec(+) < prec(*)

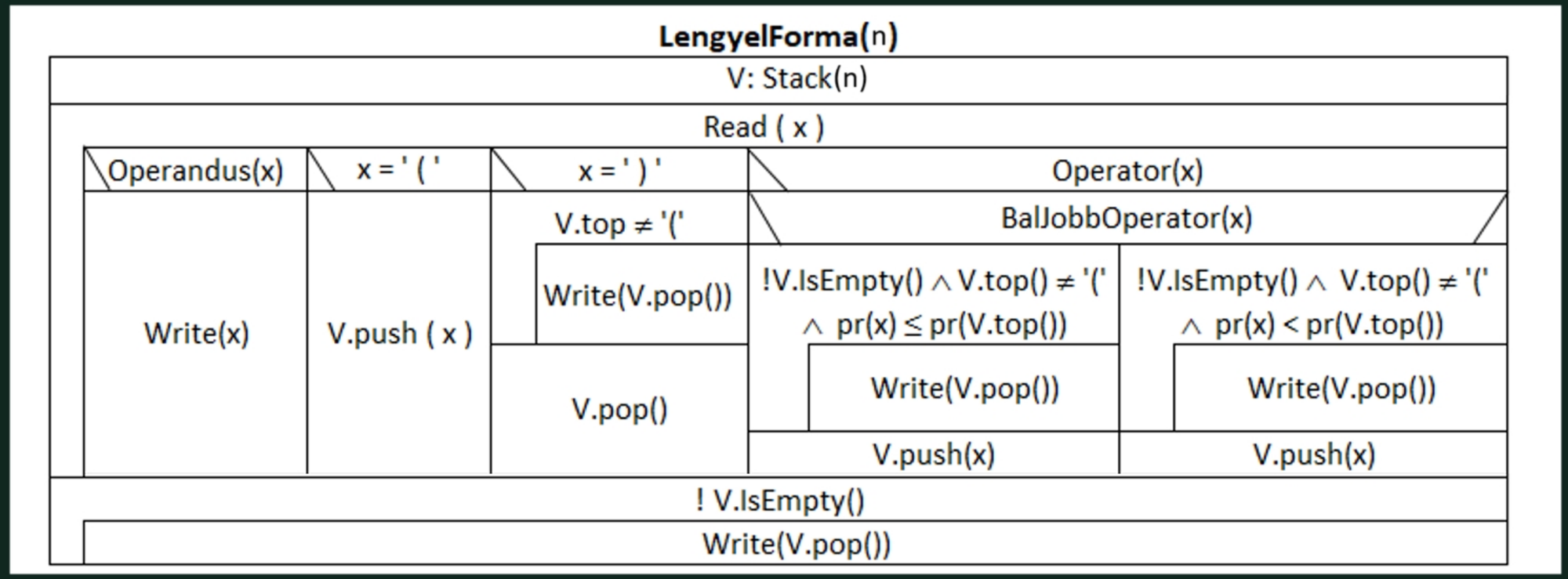
prec(+) = prec(-)

a b c d f − x + − h +



Precedencia hatása:

- Minden beolvasott műveleti jel bekerül a verembe, hogy „megvárja”, míg az operandusai kiíródnak, de előtte a veremben várakozó műveleti jelek vizsgálata történik:
- ha azonos rangú a beolvasott és a verem tetején lévő műveleti jel, kiírjuk a veremben lévőt (balról jobbra sorrend esetén) – 3. példa,
- ha a veremben magasabb prioritású művelet szerepel, mint ami bekerülne, kiírjuk – 1. példa,
- ha a verem tetején alacsonyabb rangú van, mint az olvasott, akkor bekerül a verembe – 2. példa.



$$x := a + (b - c * d) / (e - f) \hat{} g \hat{} h * i$$

prec(:=)
∧
prec(+)
prec(-)
∧
prec(*)
prec(/)
∧
prec(^)

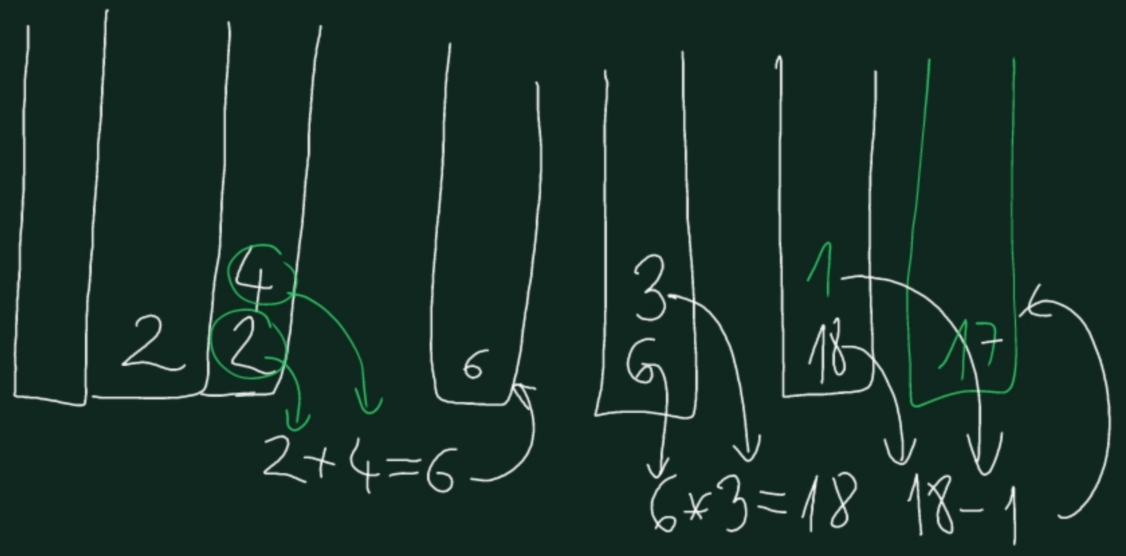
x a b c d * - e f - g h ^ ^ / i * + :=

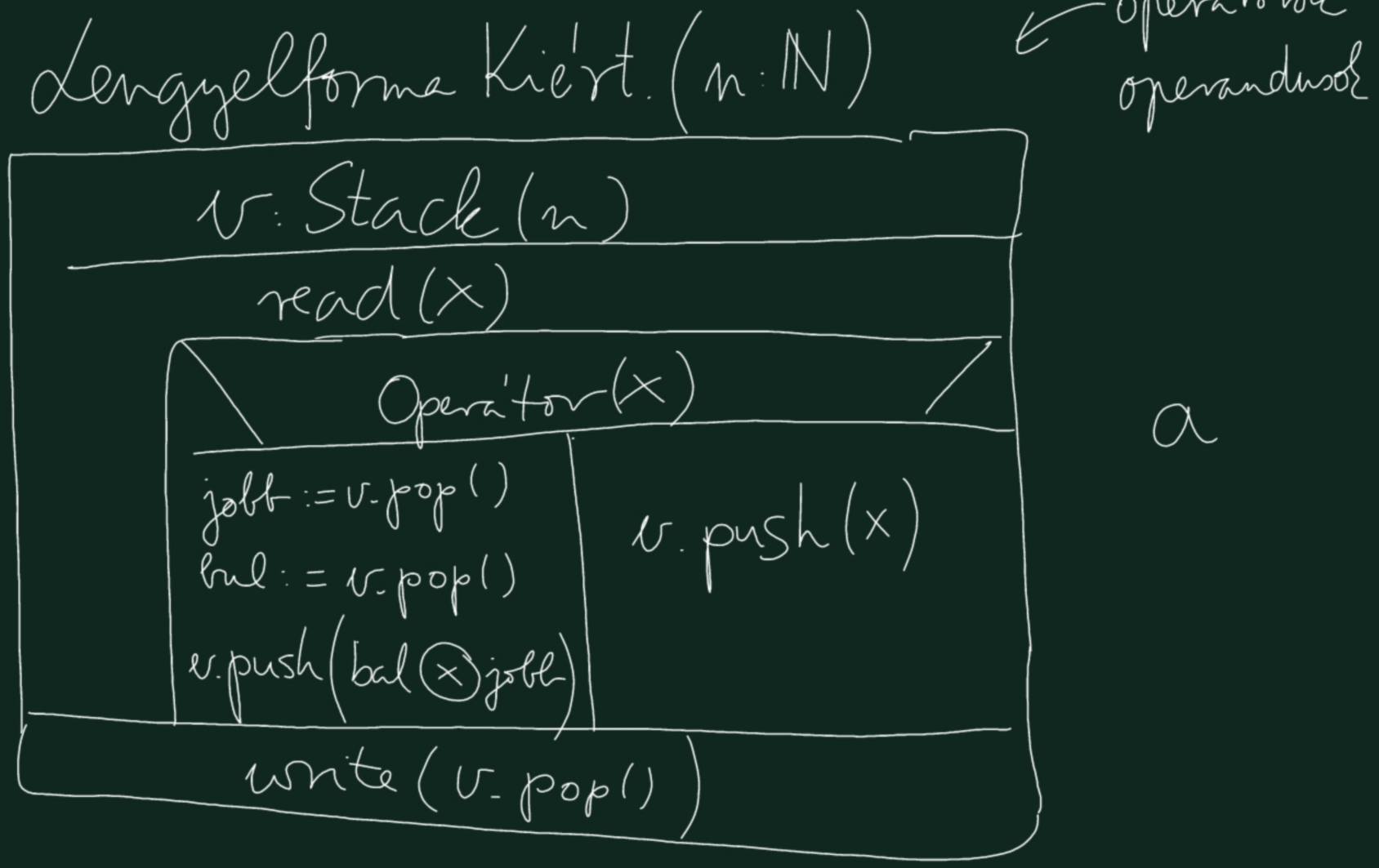
$(a+b)*c-d \longrightarrow ab+c*d-$

$a=2$
$b=4$
$c=3$
$d=1$

Kiértékelés:

$ab+c*d-$




Lengyelforma Kiért.(n:ℕ)
operátorok
operandusok
v:Stack(n)
read(x)
Operátor(x)
jobb:=v.pop()
bal:=v.pop()
v.push(bal⊗jobb)
v.push(x)
write(v.pop())
a

Teljesen zárójelezett forma

$$(((a \times b) + c) - (d / e))$$

minden művelet a két operandusával zárójelben

Feladat: algoritmus tervezése a következőkre:

(1) Teljesen és helyesen zárójelezett kifejezésből hogyan állítható elő a lengyel forma.

(2) Teljesen és helyesen zárójelezett kifejezésből hogyan értékelhető ki verem segítségével a kifejezés.

(3) Lengyel formából hogyan állíthatjuk elő a teljesen zárójelezett alakot.