# 3. Gyakorlat (2019.02.26.)

## Lengyel forma[1]

Egy aritmetikai kifejezés fostfix alakja. Jellemzői:

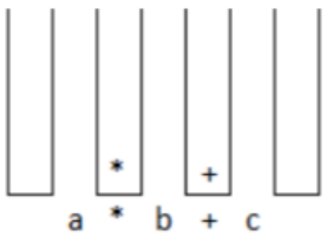
- Nincsenek benne zárójelek, a kiértékelés mégis egyértelmű, és könnyen elvégezhető
- Operandusok sorrendje nem változik, az infix kifejezéshez képes
- Operátorok sorrendje: az elvégzésük sorrendjében szerepelnek.
- Minden operátort közvetlen megelőznek az operandusai. Az operandus lehet változó, konstans,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

| infix kifejezés | lengyel forma (postfix alak) | Megjegyzés |
|---|---|---|
| a+b | ab+ | műveleti 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 |

**Feladat:** x = (a+b) * (c-d) / f ^ (g-h) + j -1 – i   kifejezés lengyel-formára hozása.
**Megoldás:** xab+cd-*fgh-^/j+1-i-=

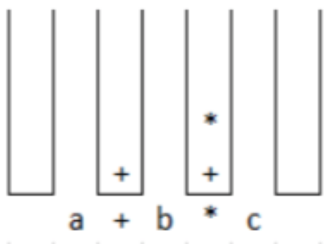
## Lengyel formára hozás verem segítségével

**1. a * b + c**



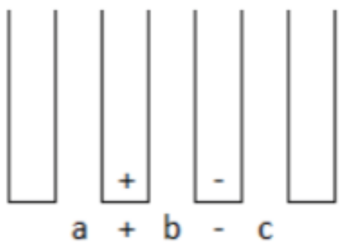

**Kimenet:** a b * c +

**2. a + b * c**




**Kimenet:** a b c * +

**3. a + b – c**




**Kimenet:** a b + c –

---

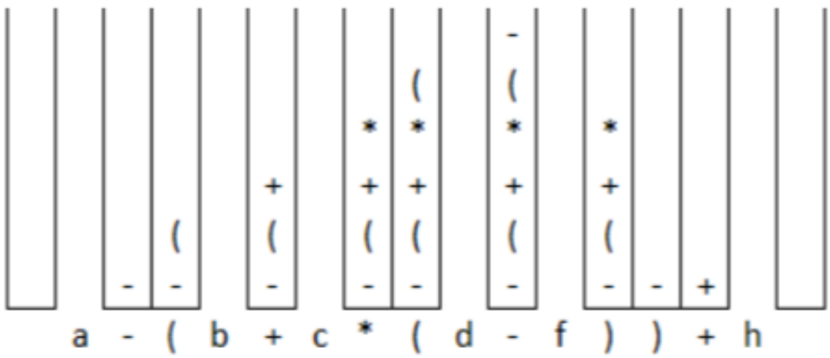
[1] – az eredeti prefix jelölés formát, **Jan Lukasiewicz** lengyel matematikus javasolta 1920-ban, később az ausztrál filozófus, **Charles Leonard Hamblin** javasolta a postfix alakot (1950), melyet emiatt „fordított lengyel formának" is szokás nevezni. (forrás: wikipedia)

**Precedencia hatása:**

- minden beolvasott műveleti jel bekerül a verembe, hogy „megvárja”, amíg 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

**Feladat:** a-(b+c*(d-f))+h kifejezés lengyel-formára hozása verem segítségével. A verem tartalmát folyamatosan tartsuk nyilván!

**Megoldás:**



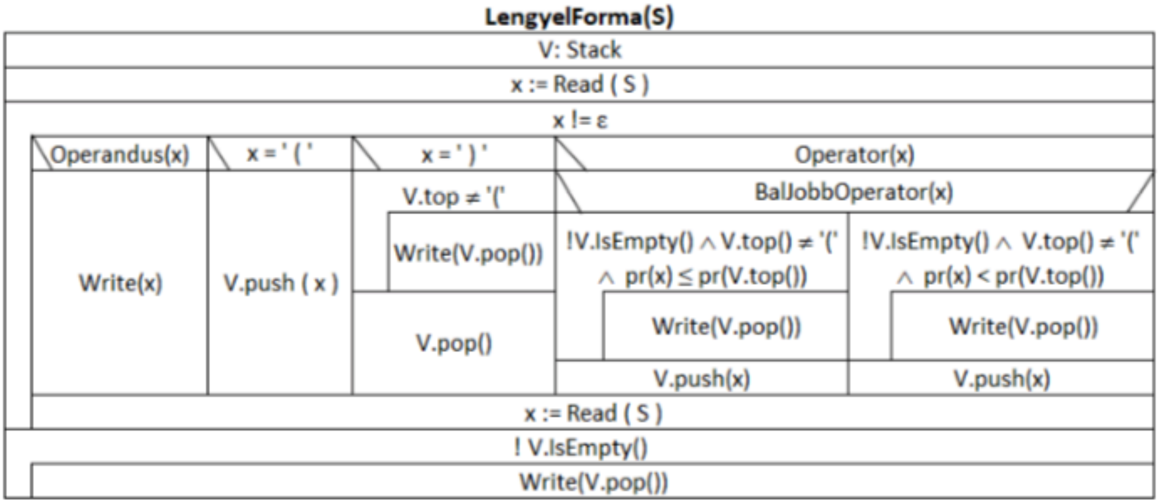

**Kimenet:** a b c d f - * + - h +

| operátor | precedencia | elvégzés |
|---|---|---|
| = | 1 | JB |
| + - | 2 | BJ |
| * / | 3 | BJ |
| ^ | 4 | JB |

- A veremben a műveleti jeleket tárolja az algoritmus : **minden műveleti jel bekerül a verembe,** hiszen várakoznia kell, amíg mindkét operandusa kiíródik a lengyel formába.
- Ha az operátor **balról jobbra** típusú ("bj operátor"), akkor mielőtt betennénk a verembe, **az összes nála nagyobb vagy egyenlő** precedenciájú műveleti jelet ki kell írni.
- Ha az operátor **jobbról balra** típusú ("jb operátor"), akkor mielőtt betennénk a verembe, **csak a nála határozottan nagyobb** precedenciájú műveleti jeleket írjuk ki. A vele egyenlőket nem.

**Algoritmus:**

Bemenet: egy helyesen zárójelezett kifejezés: S



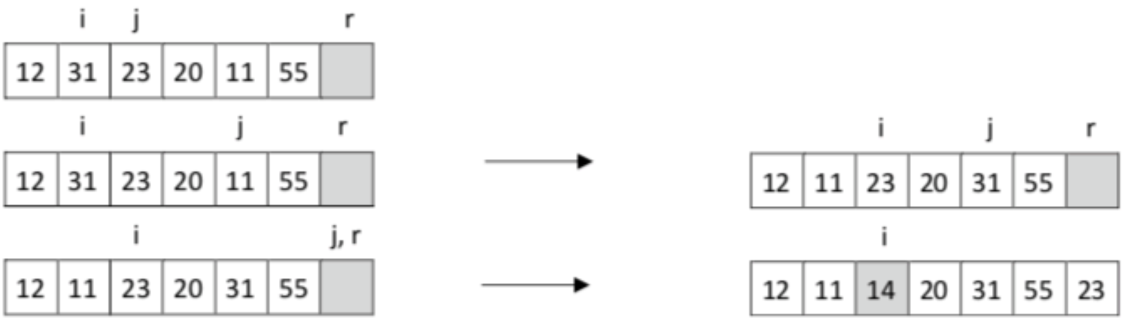

**Quick sort**

1. lépés: pivot elem véletlenszerű kiválasztása (indexe: i.). Kiírjuk a választott elemet az x segédváltozóba, majd a résztömb utolsó elemét az i-dik helyre másoljuk. A tömbrész végén egy „lyuk”-at képzünk.

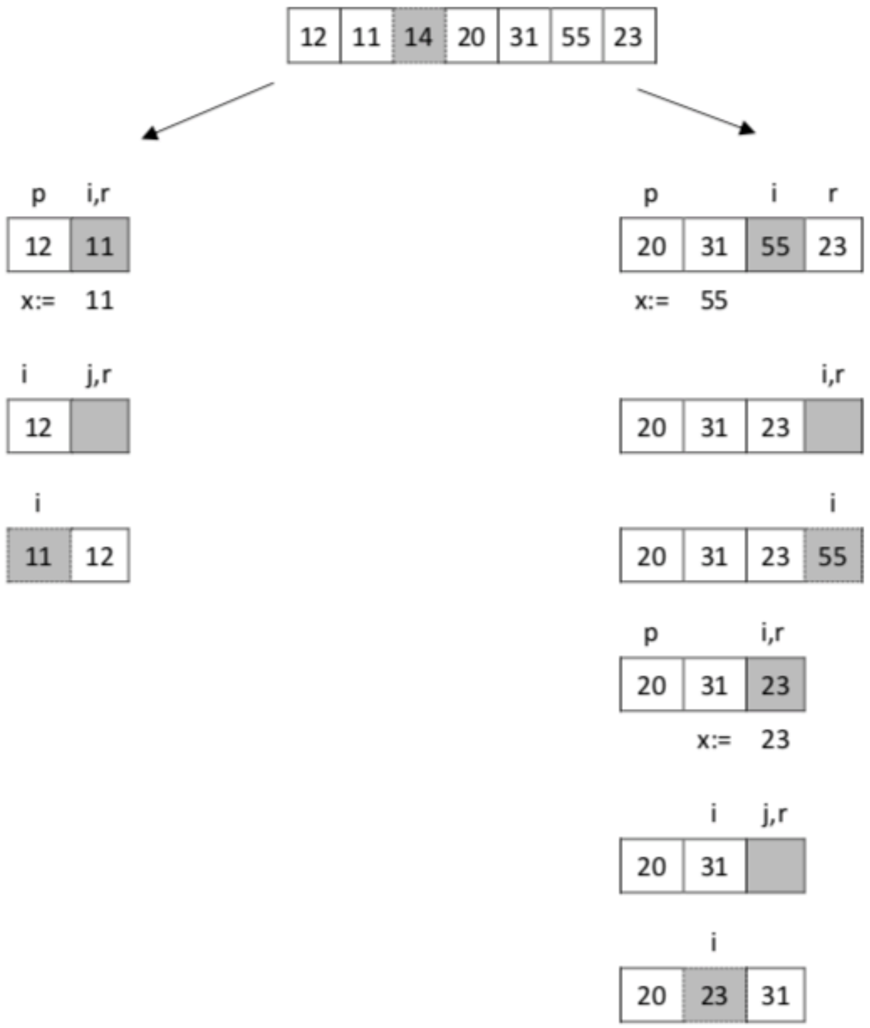

2. lépés: p-ről elindítva i-t, megkeressük az első olyan elemet, ami nagyobb, mint a pivot elem. i az algoritmus során mindig egy olyan elemre fog mutatni, amelyről tudjuk, hogy nagyobb, mint a pivot elem (ha van ilyen), és garantált, hogy előtte a pivotnál kisebbegyenlő értékek vannak. Ha nem találunk a pivot elemnél nagyobbat, akkor az azt jelenti, hogy a pivot elemnek választott elem a legnagyobb, ekkor a pivot elemet betesszük a tömb végén lévő lyukba, és vége a particionálásnak.

3. lépés: egy másik változóval, j-vel az i utáni elemről indulva lépegetünk. Ha j-vel egy pivot elemnél kisebb elemhez értünk, felcseréljük az i és j indexű elemeket. i-t ilyenkor eggyel tovább léptetjük, majd j-vel folytatjuk a tömb bejárását. A p..i-1 elemek kisebbegyenlők. i tehát mindig a vízválasztó index.

Végül j-vel elérünk az r indexig, ami a résztömb végét jelenti. r-et már nem vizsgáljuk, mert ott a „lyuk” van! Mivel i az első olyan elem indexe, ami nagyobb a pivotnál, azt kell az r indexű helyre beírni, a pivot elemet pedig betesszük az i-dik indexre. A p..r tömbrész ketté osztása az i indexnél történt, ezzel tér vissza az algoritmus.




Folytatódik a particionálás a p..i-1 és i+1..r tömbrészen:

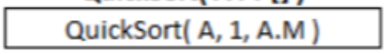



A rendezett tömb.

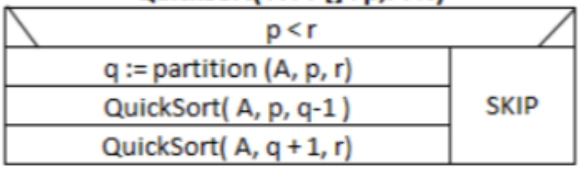
| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | 12 | 14 | 20 | 23 | 31 | 55 |

Algoritmus (előadáson szerepelt):

**QuickSort( A : T [] )**




**QuickSort( A : T [] : p,r : N)**




**partition(A : T[] ; p, r : N) : N**