3. gyakorlat

Lengyel forma[1]

Lengyel forma: egy aritmetikai kifejezés postfix alakja. Jellemzői:

- 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+	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:

A) (a + b) * (c – d) / f ^ (g – h) + j – l – i kifejezés lengyel formára hozása.

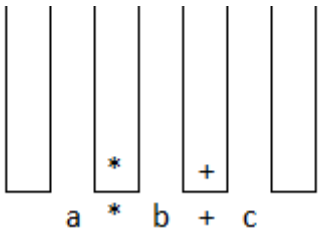
B) Értékadó operátor hatása, Hova illik az értékadó operátor?
x = (a + b) * (c – d) / f ^ (g – h) + j – l – i

Megoldás:

x a b + c d - * f g h - ^ / j + l – i - **=**

Lengyel formára hozás verem segítségével – bemutatás példákon keresztül

1. a * b + c

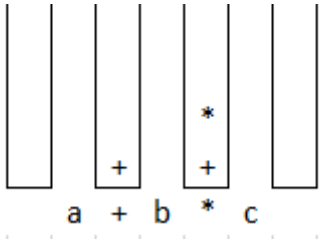


Kimenet: a b * c +

[1] Az eredeti prefix jelölési formát, **Jan Łukasiewicz** 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)

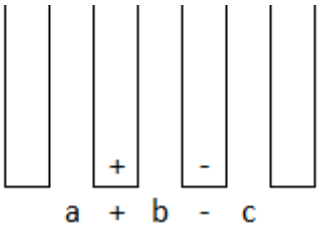
3. gyakorlat

2. a + b * c



Kimenet: a b c * +

3. a + b – c



Kimenet: a b + c –

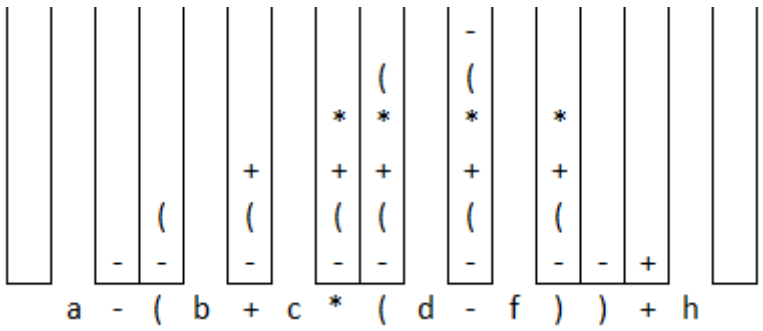
Precedencia hatása:

- 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.

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 +

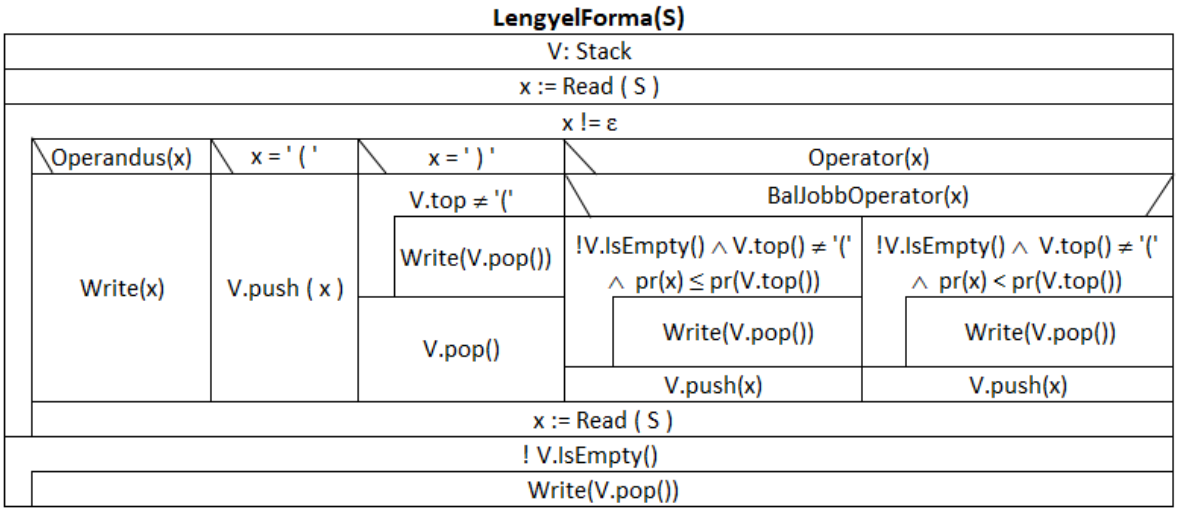
Kicsit bonyolultabb kifejezés gyakorlásra:

x = a + (- b ^ c ^ 2 + d * e) / ((f + g) * h / - k) - p * z

3. gyakorlat

Algoritmus:

Bemenet: egy helyesen zárójelezett kifejezés: S (Az operandusokat, mint szimbólumokat tesszük a verembe. Feltesszük, hogy S csak bináris operátorokat tartalmaz.)



Megemlítendő:

- Vannak jobbról balra sorrendű operátorok, ezek feldolgozása hogyan történik?
  y = x = a ^ b ^ c
- Gondoljuk meg, hogy az egy operandusú operátorok (pl. negatív előjel –a^b, vagy ++i*x) hogyan illeszthetők be a lengyel formába?
- Egyszerű függvények bevonása. pl: x = z * sin(y/w)^2
- Javasoljuk a hallgatóknak, hogy keressék meg az interneten a C++ nyelv operátorait és precedenciájukat, például: http://www.cplusplus.com/doc/tutorial/operators/

Lengyel forma kiértékelése

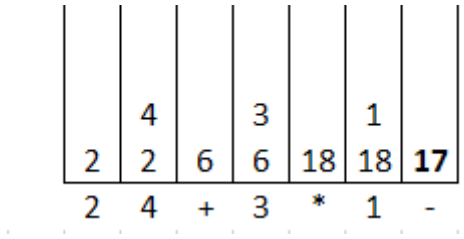
Feladat:

Hozzuk lengyel formára a következő kifejezést, majd verem segítségével értékeljük ki:
(a + b) * c – d

Lengyel forma: a b + c * d –

Tegyük fel, hogy a változók az alábbi értékkel rendelkeznek, számítsuk ki a kifejezés értékét a lengyel formából!
a = 2, b = 4, c = 3, d = 1

Kiértékelés:



Az algoritmus:

Bemenet: egy lengyel formájú kifejezés: S (Az operandusok értékét tesszük a verembe. Feltesszük, hogy S csak bináris operátorokat tartalmaz, de értékadó operátort nem.)

lengyel_kiertekeles(S)

V:Stack; x:=read(S)		
x ≠ ε		
	Operator(x)	Operandus(x)
	jobb:=V.pop()	V.push(x)
	bal:=V.pop()	
	V.push(bal ⊗ jobb)	
	x:=read(S)	
write(V.pop())		

S - lengyel formájú kifejezés

Megjegyzések az Operator(x) ághoz:

- bal ⊗ jobb jelölés: elvégezzük az x műveletet,
- feltettük, hogy az operárorok két operandusúak, de ez könnyen kiterjeszthető egy oprandusú műveletekre
- az algoritmus kiszámíthatja az értéket, vagy fordító program esetén generálhatja a kiszámítás kódját.

Javasolt házifeladatok lengyel forma témakörhöz:

Egyet érdemes feladni az (1)-(3) feladatokból, jobb csoportoknál fel lehet adni a (4)-est is.

(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.

Megoldások:

(1) Teljesen és helyesen zárójelezett kifejezésből a lengyel forma előállítása.

lengyelforma(Inp,Out)

V:Stack; x:=read(Inp)			
x ≠ ε			
x = '('	Operandus(x)	Operator(x)	x = ')'
skip	Write(Out,x)	V.push(x)	Write(Out,V.pop())
x:=read(Inp)			

Inp - egy teljesen és helyesen zárójezett aritmetikai kifejezés

Out - a kifejezés lengyel formája

3. gyakorlat

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

Kiertekeles_teljesenzarojelezettbol(Inp)

<table>
<tr><td colspan="5">V:Stack; W:Stack; z:=read(Inp)</td></tr>
<tr><td rowspan="6"></td><td colspan="4">z ≠ ε</td></tr>
<tr><td>z = '('</td><td>Operandus(z)</td><td>Operator(z)</td><td>z = ')'</td></tr>
<tr><td rowspan="4">skip</td><td rowspan="4">V.push(z)</td><td rowspan="4">W.push(z)</td><td>jobb:=V.pop()</td></tr>
<tr><td>bal:=V.pop()</td></tr>
<tr><td>x:=W.pop()</td></tr>
<tr><td>V.push(bal ⊗ jobb)</td></tr>
<tr><td colspan="4">z:=read(Inp)</td></tr>
<tr><td colspan="5">Write(V.pop())</td></tr>
</table>

V - operandusokat tároló verem

W - operátorokat tároló verem

Inp - egy teljesen és helyesen zárójelezett kifejezés.

z=')' ágra ugyanazok a megjegyzések vonatkoznak, mint a lengyel forma kiértékelő algoritmusra