

Veri Yapıları ve Algoritmalar

Infix, Postfix, Prefix Gösterimler

Giriş

- Matematiksel işlemlerin genel gösterim şekli
 - $a+b$
 - Bu ifade de a ve b veri/değişken(***operand***), $+$ ise yapılacak işlem operatörüdür (***operand***).
 - Örneğin bu gösterim türü *infix* olarak adlandırılır.
- Daha karmaşık ifadeler?
 - $a+b/c$
 - $(a+b)/c$
 - $a+(b/c)$

Infix, Postfix, Prefix Gösterimleri

- Infix
 - operand *operatör* operand
 - $a+b$
 - İşlem öncelik sırasına göre ve soldan sağa ilerler.
- Prefix
 - *operator* operand operand
 - $+ab$
 - Öncelik yoktur ve sağdan sola doğru ilerler
- Postfix
 - operand operand *operatör*
 - $ab+$
 - Öncelik yoktur ve soldan sağa ilerler.

Karşılaştırma

- infix
 - Basit ve anlaşılırdır. Kimin için?
 - işlem öncelikleri için parantezler kullanılır.
- Postfix/Prefix
 - Parantez kullanımına gerek yoktur!
 - Bir çok derleyici infix notasyonunda ki ifadeleri postfix/prefix e çevirerek saklar.

İşlem Önceliği

- Öncelik sırası
 - Parantez
 - Üs
 - Çarpma/Bölme
 - Toplama/Çıkarma
- Parantez yok ise aynı önceliğe sahip işlemler soldan sağa doğru yapılır.
- ***Üs alma işleminde sağdan sola doğru yapılır!***
 - $x+y-z=(x+y)-z$
 - $z^y^x = z^{(y^x)}$

Örnek: Üs alma işlem önceliği

2^3^4 işlemini ele alalım

- $(2^3)^4 = 8^4 = 4096$
- $2^{(3^4)} = 2^{81} = 2417851639229258349412352$
- Python ortamında deneyin!
`print(2**3**4)`

Örnek

$2+3*4$ işlemini ele alalım

- $(2+3)*4=20$
- $2+(3*4)=14$
- infix gösterim
 $2+(3*4)=14$

Örnek: Prefix/Postfix gösterim

Infix: $2+3*4$

Prefix

$+2*34$

Postfix

$234*+$

Prefix gösterimde operatörler **önce** yazılır, işlem sağdan sola gerçekleşir.

Postfix gösterimde operatörler **sonra** yazılır, işlem **soldan sağa** gerçekleşir.

İki operand bir operatör kuralını unutmayalım!

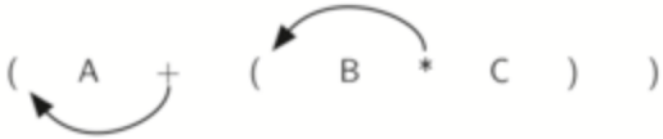
Tam Parantez

- Infix, postfix/prefix dönüşümlerinde tam parantez kullanımı işimizi kolaylaştırır.
- Nedir tam parantez?
- $(a+b)*c$ anlaşılır bir işlemdir.
- $((a+b)*c)$ dönüşüm işlerinde daha anlaşılır bir işlemdir.
- **operand operator operand = new operand**
- Üstteki ifadeye göre her işlem sonucu yeni bir değişken olacaktır ve bu değişken başka bir operand ile yeni işleme tabi tutulacaksa parantez kullanılmalıdır.

Dönüşümler

- Infix -> Prefix

- Operatörü kendi operandlarının soluna taşı ve parantezi kaldır.



- Infix -> Postfix

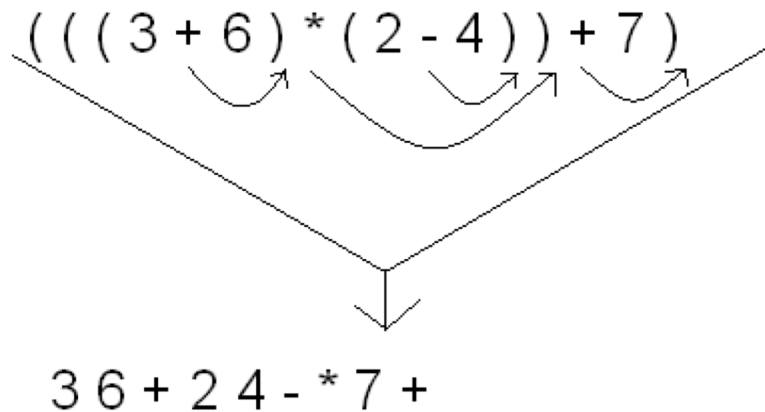
- Operatörü kendi operandlarının sağına taşı ve parantezi kaldır.



<http://interactivepython.org/runestone/static/pythonds/BasicDS/InfixPrefixandPostfixExpressions.html>

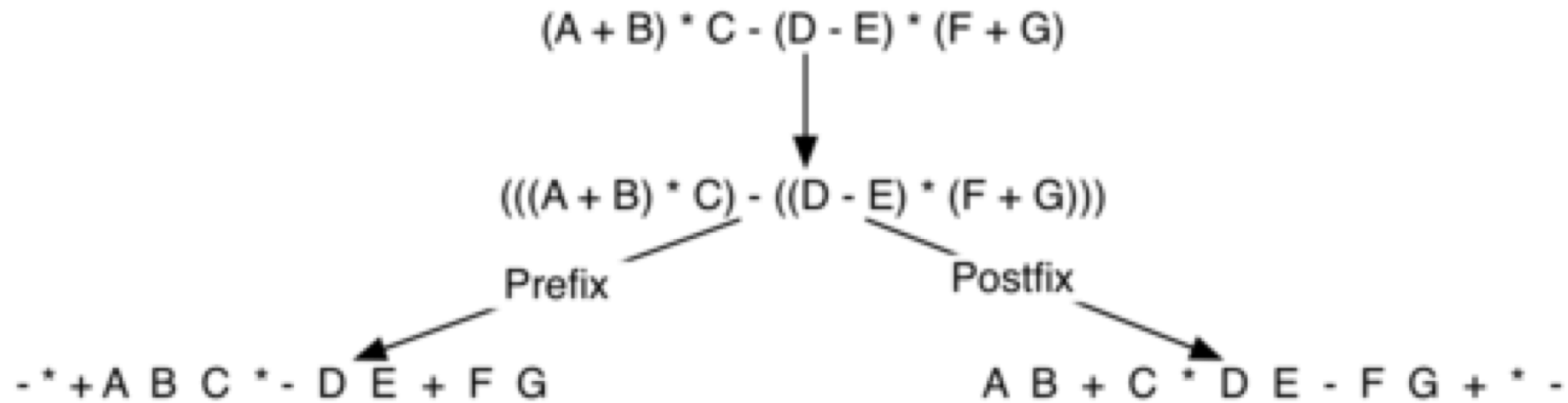
Tam Parantez

- Tam parantez kullanımının dönüşüm işlemini kolaylaştırdığına dikkat edin!



<https://olimex.files.wordpress.com/2013/07/in-post.gif>

Örnek dönüşüm



http://interactivepython.org/runestone/static/pythonds/_images/complexmove.png

Örnekler

- Aşağıdaki örnekleri inceleyin.

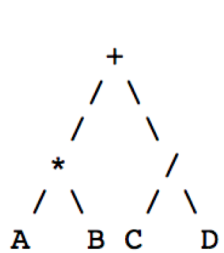
Infix	Postfix	Prefix
$((A * B) + (C / D))$	$((AB *) (CD /) +)$	$(+ (* AB) (/ CD))$
$((A * (B + C)) / D)$	$((A (BC +) *) D /)$	$(/ (* A (+ BC)) D)$
$(A * (B + (C / D)))$	$(A (B (CD /) +) *)$	$(* A (+ B (/ CD)))$

https://mcalkanyurekblog.files.wordpress.com/2017/11/infix_postfix_parantezli.png?w=656

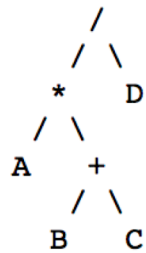
Örnekler

- Dönüşüm işleminde ağaç kullanılabilir, kökler operatör, yapraklar operand.

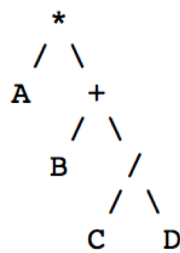
Infix	Postfix	Prefix
$((A * B) + (C / D))$	$((AB*)(CD/)+)$	$(+(*AB)(/CD))$
$((A * (B + C)) / D)$	$((A(BC+)*D/)$	$(/(*A(+BC))D)$
$(A * (B + (C / D)))$	$(A(B(CD/)+)*)$	$(*A(+B(/CD)))$



$((A*B)+(C/D))$



$((A*(B+C))/D)$



$(A*(B+(C/D)))$

<http://www.cs.man.ac.uk/~pjj/cs212/fix.html>

Örnekler

- Aşağıdaki örnekleri inceleyin.

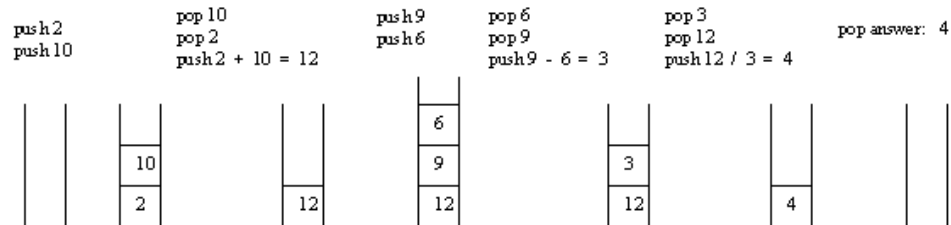
Infix	Postfix	Prefix
$A+B-C$	$AB+C-$	$-+ABC$
$(A+B)*(C-D)$	$AB+CD-*$	$*+AB-CD$
$A^B*C-D+E/F/(G+H)$	$AB^C*D-EF/GH+/+$	$+-*^{\wedge}ABCD//EF+GH$
$((A+B)*C-(D-E))^{(F+G)}$	$AB+C*DE-FG+^{\wedge}$	$^{\wedge}-*+ABC-DE+FG$
$A-B/(C*D^E)$	$ABCDE^{\wedge}*/-$	$-A/B*C^{\wedge}DE$

<http://www.barankisa.com/wp-content/uploads/2018/01/%C4%B0LKKK.png>

Yığıt Kullanımı

- LIFO: Last in First Out
- Prefix/Postfix işlemler yığıt(stcak kullanılarak yapılabilir)
- **İki operand çekilir, işlem yapılır tekrar yığita atılır.**
- Yığıt boşalana kadar işlem devam eder.

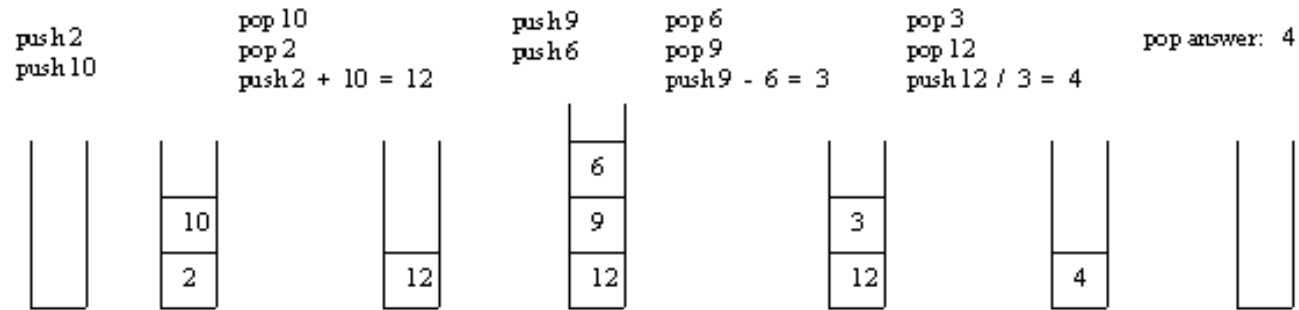
2 10 + 9 6 - /



Örnek

- infix işlem $(2+10)/(9-6)$
- Postfix $2\ 10\ +\ 9\ 6\ -\ /\$
- Yığıt işleyişine dikkat edelim.

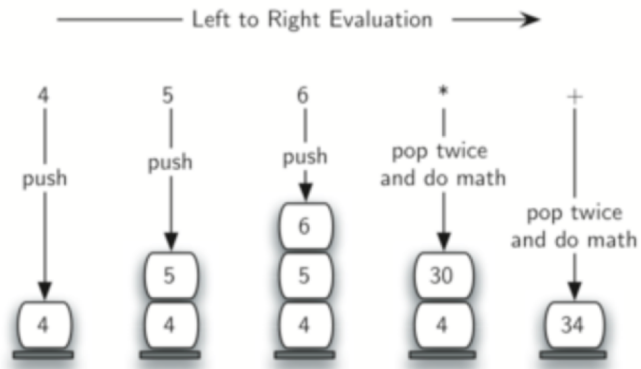
$2\ 10\ +\ 9\ 6\ -\ /\$



<https://www.thecrazyprogrammer.com/wp-content/uploads/2014/02/Evaluation-of-a-postfix-expression-using-a-stack.gif>

Örnek

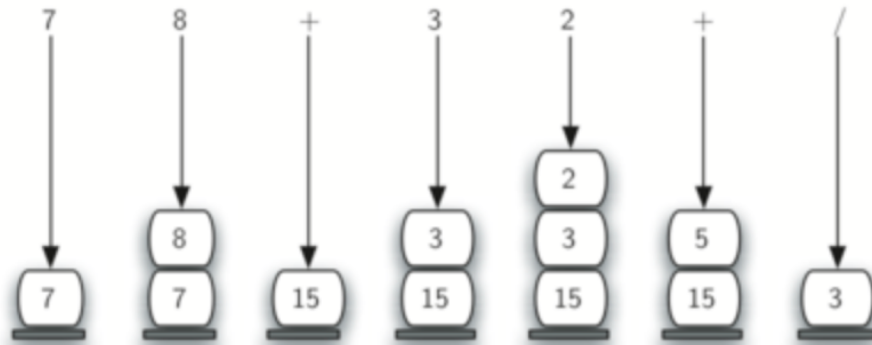
- infix işlem $(4+(5*6))$
- Postfix $5\ 6\ *\ +$
- Yığıt işleyişine dikkat edelim.



<https://www.thecrazyprogrammer.com/wp-content/uploads/2014/02/Evaluation-of-a-postfix-expression-using-a-stack.gif>

Örnek

- infix işlem $((7+8)/(3+2))$
- Postfix $7\ 8\ +\ 3\ 2\ +\ /$
- Yığıt işleyişine dikkat edelim.



<https://www.thecrazyprogrammer.com/wp-content/uploads/2014/02/Evaluation-of-a-postfix-expression-using-a-stack.gif>

Pratik için

- infix/prefix/postfix dönüşümleri ve stack için internet ortamında birçok online tool bulunmakta.

<https://raj457036.github.io/Simple-Tools/prefixAndPostfixConvertor.html>

Ödev

- 1) infix olarak verilen bir işlemi (string bir ifade) postfix ve prefixe çeviren program yazınız. Program işlem sonucunun doğruluğunu teyit etmelidir.
- 2) String olarak girilen postfix işlem sonucunu yığıt yapısı ile bulan programı nesne tabanlı olarak kodlayınız.