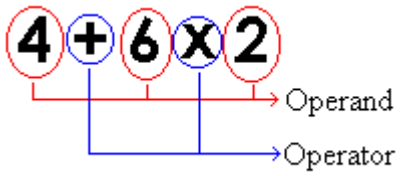
INFIX, PREFIX DAN POSTFIX

• PENDAHULUAN

Salah satu kegunaan stack adalah untuk mengubah notasi infix menjadi prefix ataupun postfix. Sebelum kita lihat yang dimaksud dengan infix, prefix dan postfix, ada baiknya mengenal istilah operand dan operator dahulu.

Apa itu Operand dan Operator ?



Lalu apa sih Infix, Prefix, Postfix tersebut ?

Infix, Prefix ataupun Postfix adalah bentuk penulisan operasi matematika, bedanya :
Infix = Operator diletakkan di antara Operand
Prefix = Operator diletakkan di depan Operand
Postfix / **Su**fix = Operator diletakkan di belakang Operand

Contoh :

Infix	Prefix	Postfix
A + B x C	x + A B C	A B C x +
D / E – F	- / D E F	D E / F -

Bingung kenapa Prefix atau Postfix-nya bisa seperti itu?

Mengapa sih mesti ada notasi Prefix atau Postfix? Kenapa nggak gunakan infix aja, seperti yang sudah kita pelajari sejak zaman TK?

Karena infix memiliki beberapa kekurangan, yaitu :

1. Urutan pengerjaan tidak berdasarkan letak kiri atau kananya, tetapi berdasarkan *precedence*-nya
   Contoh : 3 + 4 x 2
   3 + 4 x 2 , maka urutan pengerjaan adalah 4 x 2 dahulu.
   3 + 8 , baru hasilnya ditambah 3
   11
   Urutan **precedence** (dari prioritas tertinggi) adalah sebagai berikut :
   1. Pemangkatan
   2. Perkalian dan Pembagian
   3. Penjumlahan dan Pengurangan.

   Kecuali kalau ada tanda kurung.

2. Menggunakan tanda kurung. Betul, infix bisa menggunakan tanda kurung. Repotnya, si tanda kurung ini bisa ngacak-ngacak urutan precedence.
   Contoh : Tanpa penggunaan tanda kurung :
   9 – 5 – 3
   9 – 5 – 3 , maka urutan pengerjaan adalah 9 - 5 dahulu.
   4 – 3
   1

   Bandingkan dengan penggunaan tanda kurung berikut :
   9 – (5 – 3)
   9 – (5 – 3) , maka urutan pengerjaan adalah 5 – 3 dahulu.
   9 – 2
   7

3. Jika suatu program akan mengevaluasi (mencari hasil) suatu infix, maka komputer perlu men-scan berulang-ulang mencari urutan pengerjaannya dahulu.
   Contoh : 7 + 4 x 2 – 6 / 3

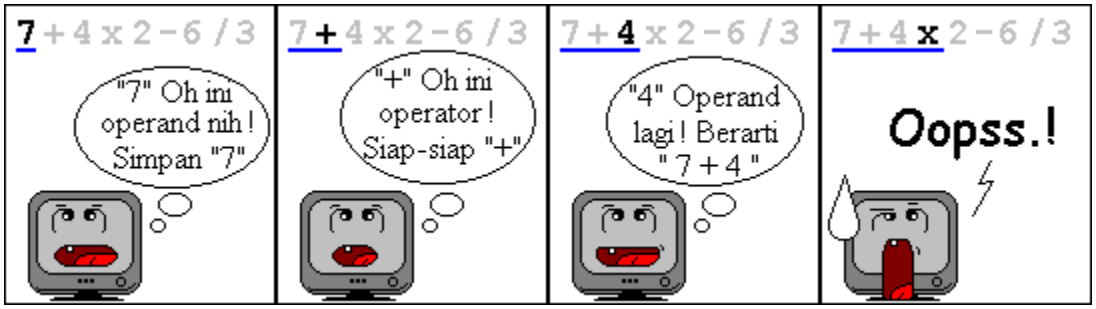
   Jika kita diminta untuk menghitung soal seperti itu, maka kita tahu bahwa yang pertama kali harus kita kerjakan adalah 4 x 2. Lalu 6 / 3 dsb, seperti langkah-langkah berikut :

   7 + 4 x 2 – 6 / 3
   7 + 8 – 6 / 3
   7 + 8 – 2
   15 – 2
   13 Bingung pada tahapan ini?

   Jangan lupa bahwa pengerjaan suatu soal tergantung pada urutan precedence-nya.

   Masalahnya, si komputer tidak bisa membaca keseluruhan soal sekaligus. Komputer hanya bisa men-scan soal satu per satu operand atau operator. Sehingga untuk mengetahui mana yang harus dikerjakan duluan, komputer harus men-scan keseluruhan soalnya dulu. Jadi langkah-langkah si komputer dalam mengerjakan soal infix seperti berikut:
   1. Cari precedence tertinggi dengan men-scan kiri ke kanan keseluruhan soal.
   2. Hitung nilai operator dengan precedence tertinggi tersebut.
   3. Ulangi lagi dari langkah 1, sampai semua operator selesai dikerjakan.

   Jika si komputer yang tidak men-scan keseluruhan soalnya dulu, maka bisa salah hasilnya.



- **KONVERSI**

Konversi Infix ke Postfix Manual

Langkah-langkahnya :

1. Cari operator yang memiliki precedence tertinggi.
2. Letakkan operator tsb di belakang operand-operandnya.
3. Ulangi terus sampai bosan, eh salah, sampai selesai.

Contoh:

A + B – C x D ^ E / F , ”D ^ E” maksudnya tuh D pangkat E.

A + B – C x D ^ E / F , pangkat memiliki precedence tertinggi
A + B – C x D E ^ / F , taruh ^ di belakang D dan E
A + B – C x D E ^ / F , x (kali) dan / (bagi) memiliki precedence sama tapi x di kiri
A + B – C D E ^ x / F , taruh x di belakang
A + B – C D E ^ x / F , dsb..., pelajari saja dulu.
A + B – C D E ^ x F /
A + B – C D E ^ x F /
A B + – C D E ^ x F /
A B + – C D E ^ x F /
A B + C D E ^ x F / – , inilah bentuk Postfix-nya.

Konversi Infix ke Postfix Menggunakan Stack

Bahan-bahan yang dibutuhkan :

1 bh stack, misal bernama “Stack”.
1 bh rangkaian soal, misal bernama “Infix”.
1 bh variabel untuk penampung, misal bernama “Postfix”.
Pikiran yang jernih secukupnya.
Kemauan mencoba sesuai selera.

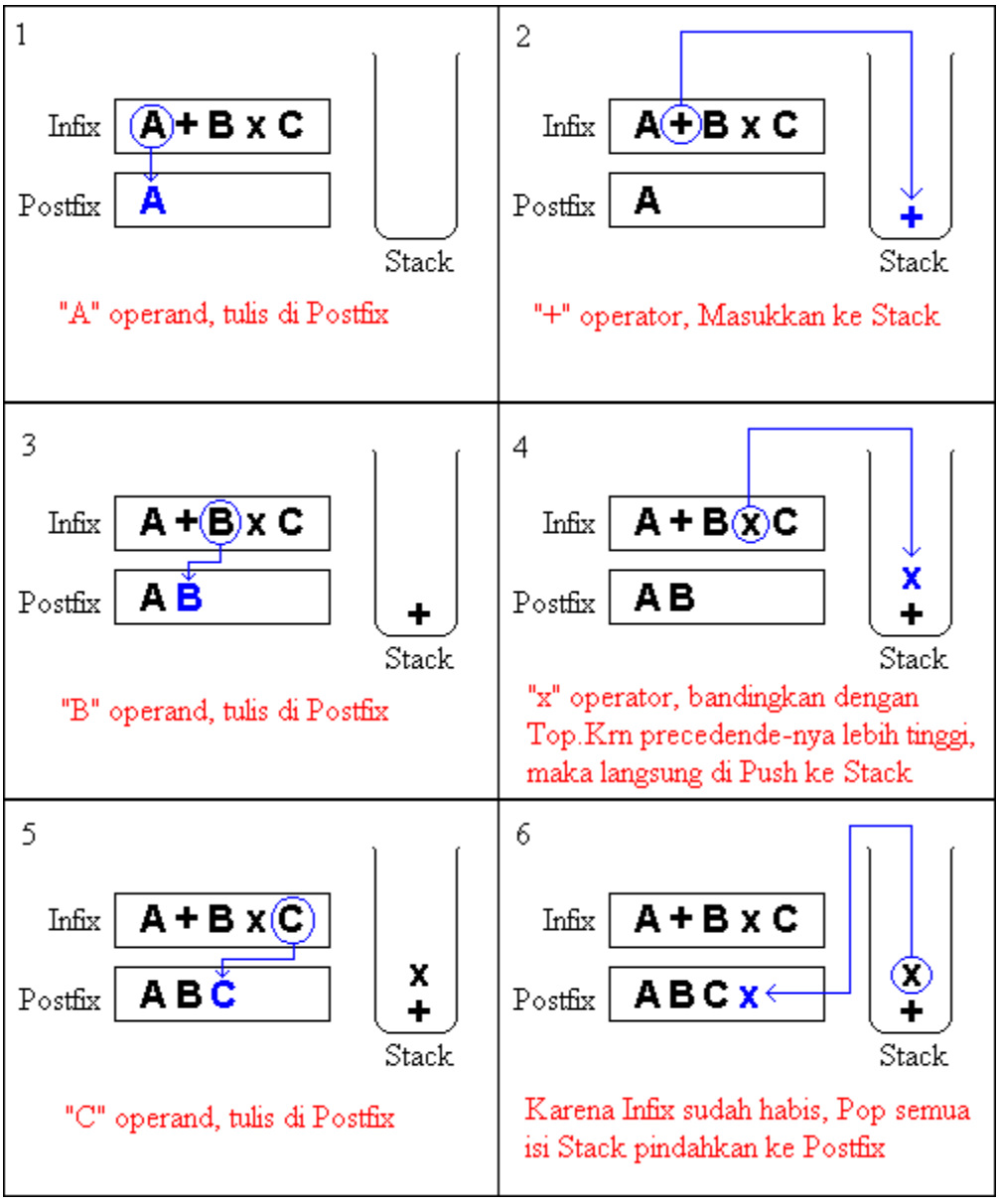
Cara pembuatan :

1. Scan Infix dari kiri ke kanan.
2. Jika berupa operand, maka tulis di Postfix.
3. Jika berupa operator, maka bandingkan operator NEW tsb dgn TOP pada Stack :

   a. WHILE precedence TOP >= NEW, maka POP Stack pindahkan ke Postfix.
   b. Lalu Push NEW ke dalam Stack.
4. Jika berupa "(", maka Push "(" ke Stack.
5. Jika berupa ")", maka Pop Stack pindahkan ke Postfix sampai ketemu "(".
6. Ulangi terus dari langkah 1 sampai seluruh Infix sudah di-scan.
7. POP semua isi Stack, pindahkan ke Postfix.

Perlu diingat !! tanda kurung "(" ataupun ")" tidak dimasukkan ke Postfix.

Contoh : A + B x C



Infix, Prefix, Postfix

Contoh Lagi : A ^ B / (C – D)

	Infix	Postfix	Stack
1.	A ^ B / (C – D)	A	
2.	A ^ B / (C – D)	A	^
3.	A ^ B / (C – D)	A B	^
4.	A ^ B / (C – D)	A B ^	/
5.	A ^ B / (C – D)	A B ^	/ (
6.	A ^ B / (C – D)	A B ^ C	/ (
7.	A ^ B / (C – D)	A B ^ C	/ (–
8.	A ^ B / (C – D)	A B ^ C D	/ (–
9.	A ^ B / (C – D)	A B ^ C D –	/
10.	A ^ B / (C – D)	A B ^ C D – /	

Keterangan :

- Tanda kurung "(" dan ")", dapat dianggap tidak memiliki precedence, sehingga pada langkah ke-7, operator "–" tidak perlu dibandingkan lagi dengan "(" dan langsung di Push ke Stack.
- Pada langkah ke-8, tanda ")" dibaca dari Infix, maka Stack di Pop terus sampai ketemu tanda "(". Sehingga pada contoh di atas operator "–" di Pop dan dipindahkan ke Postfix.

Konversi Infix ke Prefix Manual

Langkah-langkahnya (mirip dengan Infix ☐ Postfix loh ! dicek aja deh) :

1. Cari operator yang memiliki precedence tertinggi.
2. Letakkan operator tsb di depan operand-operandnya.
3. Ulangi lagi.

Contoh:

A + B – C x D ^ E / F , ”D ^ E” lagi-lagi maksudnya D pangkat E.

A + B – C x D ^ E / F , pangkat memiliki precedence tertinggi
A + B – C x ^ D E / F , taruh ^ di depan D dan E
A + B – C x ^ D E / F , x (kali) dan / (bagi) memiliki precedence sama tapi x di kiri
A + B – x C ^ D E / F , taruh x di belakang
A + B – x C ^ D E / F , dsb..., pelajari lagi saja dulu.
A + B – / x C ^ D E F
A + B – / x C ^ D E F
+ A B – / x C ^ D E F
+ A B – / x C ^ D E F
– + A B / x C ^ D E F , inilah bentuk Prefix-nya.

Miripkan langkah-langkahnya dengan Infix ☐ Postfix yang tadi. Bahkan karena miripnya, sehingga saya menggunakan cara copy-paste aja untuk contoh Prefix di atas ☐ hehehe.

Konversi Infix Ke Prefix Menggunakan Stack

Kali ini kita menggunakan 2 Stack, yang satu untuk menampung operand (saya sebut aja namanya Stack “Pre”) dan yang satunya lagi untuk menampung operator (yang ini saya sebut Stack “Opr” deh).

Langkah – langkah :

1. Scan Infix dari kanan ke kiri.
2. Jika berupa operand, maka Push ke Stack “Pre”.
3. Jika berupa operator, maka bandingkan operator NEW tersebut dengan TOP pada Stack “Opr”:
    a. WHILE precedence TOP > NEW, maka POP Stack “Opr” pindahkan ke Stack “Pre”.
    b. Lalu Push NEW ke dalam Stack “Opr”.
4. Jika berupa “)“, maka Push “)“ ke Stack “Opr”.
5. Jika berupa “(”, maka Pop Stack “Opr” pindahkan ke stack “Pre” sampai ketemu “)“.
6. Ulangi terus dari langkah 1 sampai seluruh Infix sudah di-scan.
7. POP semua isi Stack “Opr”, pindahkan ke Stack “Pre”.
8. POP semua isi Stack “Pre”, pindahkan ke Prefix.

Contoh yang mirip Postfix tadi : A ^ B / (C – D)

	Infix	Stack "Pre"	Stack "Opr"
1.	A ^ B / (C – D)		)
2.	A ^ B / (C – D)	D	)
3.	A ^ B / (C – D)	D	) –
4.	A ^ B / (C – D)	D C	) –
5.	A ^ B / (C – D)	D C –	
6.	A ^ B / (C – D)	D C –	/
7.	A ^ B / (C – D)	D C – B	/
8.	A ^ B / (C – D)	D C – B	/ ^
9.	A ^ B / (C – D)	D C – B A	/ ^
10.	A ^ B / (C – D)	D C – B A ^	/
11.	A ^ B / (C – D)	D C – B A ^ /	
12.	Prefix-nya menjadi / ^ A B – C D		

Keterangan :

- Setelah Stack "Opr" dikosongkan. Jangan lupa untuk memindahkan isi Stack "Pre" ke Prefix. Sehingga urutan peletakkan operand pada hasil akhirnya tetap sama **(coba diperhatikan, ternyata sewaktu bentuk Infix diubah ke Postfix ataupun Prefix, urutan letak operand-nya tetap sama !!!)**.

• EVALUASI

Yang dimaksud dengan "Evaluasi" disini adalah mencari nilai akhir dari suatu notasi. Dengan kata lain, disuruh ngitung hasilnya.
Contoh : Berapa hasil 3 + 4 ?
Jawab : 7 (bingung nilai 7 dapat dari mana? Coab dibuka buku matematika kelas 1 SD).
Yah udah cuma begitu aja.

Evaluasi Postfix Manual

Langkah-langkahnya :

1. Scan Postfix dari kiri ke kanan.
2. Jika berupa operand, cuekin dulu aja.
3. Jika berupa operator, ambil 2 operand sebelumnya (yang tadi sempet kita cuekin di sebelah kiri), lakukan perhitungan, lalu simpan lagi berupa operand.
4. Begitu seterusnya sampai ujung kanan Postfix.

Contoh :

Postfix : 7 6 5 x 3 2 ^ – +

7 6 5 x 3 2 ^ – + , scan terus sampai ketemu operator pertama.
7 6 5 x 3 2 ^ – + , hitung 6 x 5.

7 30 3 2 ^ – + , scan lagi cari operator berikutnya.
7 30 3 2 ^ – + , hitung 3 pangkat 2.
7 30 9 – + , scan lagi cari operator berikutnya.
7 30 9 – + , hitung 30 – 9.
7 21 + , scan lagi.
7 21 + , hitung 7 + 24.
28 , selesai.

Evaluasi Postfix Menggunakan Stack

Langkah-langkahnya :

1. Scan Postfix dari kiri ke kanan.
2. Jika berupa Operand, masukkan ke Stack.
3. Jika berupa Operator, Pop Stack 2 kali (ambil 2 operand), hitung hasilnya, lalu Push lagi ke dalam Stack.
4. Ulangi lagi sampai ujung kanan Postfix.

Evaluasi Prefix Manual

Langkah-langkahnya idem, sama kaya Postfix, tapi arah scannya dari kanan ke kiri.

Contoh :

Prefix : + 7 – x 6 5 ^ 3 2 (soalnya sama nih sama soal Postfix tadi)

\+ 7 – x 6 5 ^ 3 2 , scan kanan ke kiri sampai ketemu operator.
\+ 7 – x 6 5 ^ 3 2 , hitung 3 pangkat 2.
\+ 7 – x 6 5 9 , selanjutnya silahkan pelajari sendiri dulu.
\+ 7 – x 6 5 9
\+ 7 – 30 9
\+ 7 – 30 9
\+ 7 21
\+ 7 21
28

Evaluasi Postfix Menggunakan Stack

Langkah-langkahnya :

1. Scan Postfix dari kanan ke kiri.
2. Jika berupa Operand, masukkan ke Stack.
3. Jika berupa Operator, Pop Stack 2 kali (ambil 2 operand), hitung hasilnya, lalu Push lagi ke dalam Stack.
4. Ulangi lagi sampai ujung kanan Postfix.

Karena caranya mirip, lagi-lagi saya gunakan copy-paste ☐.

--- SELAMAT MENCOBA ---