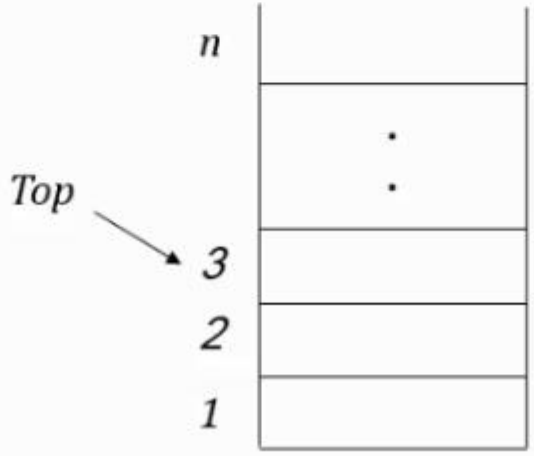
ساختمان داده ها والگوریتم ها

فصل سوم

پشته(Stack)

تعریف پشته(Stack)

به لیستی از داده ها که عمل حذف (POP) و درج (PUSH) در آن از یک طرف (بالای پشته) به کمک اشاره گری به نام "Top" انجام می گیرد پشته گویند.



تعریف پشته(Stack)

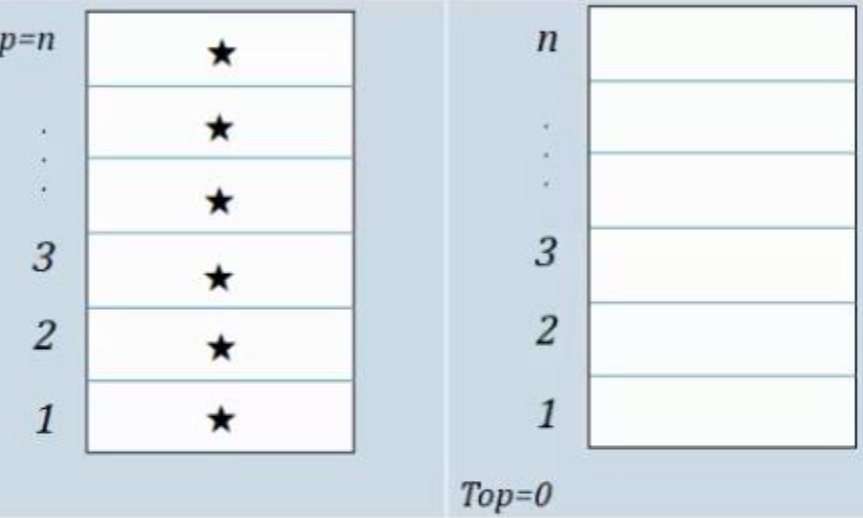
حالت Top:

۱- اشاره به آخرین خانه پر

۲- اشاره به اولین خانه خالی

★ همیشه به طور پیش فرض اشاره گر Top به آخرین خانه پر اشاره می کند.

تعریف پشته(Stack)

وضعیت Top	پشته خالی (Empty)	پشته پر (Full)
آخرین خانه پر		
اولین خانه خالی		

خروجی های مجاز

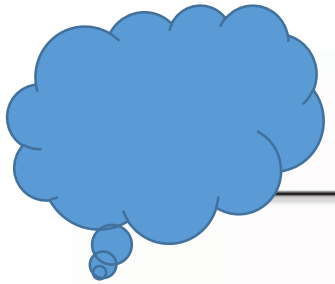
در یک پشته nتایی در صورتی که داده ها به ترتیب a_1 و a_2 و a_3 و ... و a_n به ترتیب وارد پشته شوند،

خروجی های مجاز به صورت زیر خواهند بود:

۱) هر داده به محض ورود می تواند از پشته خارج شود.

۲) هر گاه بخواهیم داده ای را در خروجی ببینیم که هنوز وارد پشته نشده است، باید داده ها را پشت سرهم وارد پشته کنیم تا به داده مورد نظر برسیم، سپس آن را خارج کنیم.

۳) هرگاه بخواهیم داده ای را در خروجی ببینیم که پایین پشته است و عنصر بالای پشته نیست، این عمل غیر مجاز است.



پیاده سازی پشته

برای ذخیره سازی ساختار یک پشته می توان یکی از ۲ ساختار زیر را در نظر گرفت:

۱- **آرایه:** که ساختاری ایستا و محدود است.

۲- **لیست پیوندی:** که ساختاری پویا و متناسب با داده هاست.

الگوریتم Push

وضعیت Top: اشاره به آخرین خانه پر

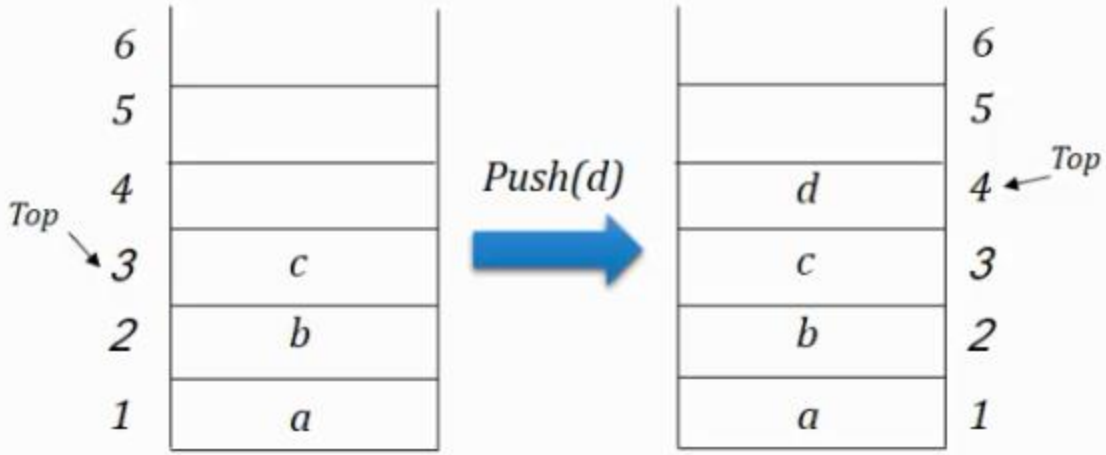
```
Void Push ( type item ){
If ( top == n )
    cout<<“Stack is FULL”;
Else{
    top+=1;
    stack[top] = item;}
}
```

وضعیت Top: اشاره به اولین خانه خالی

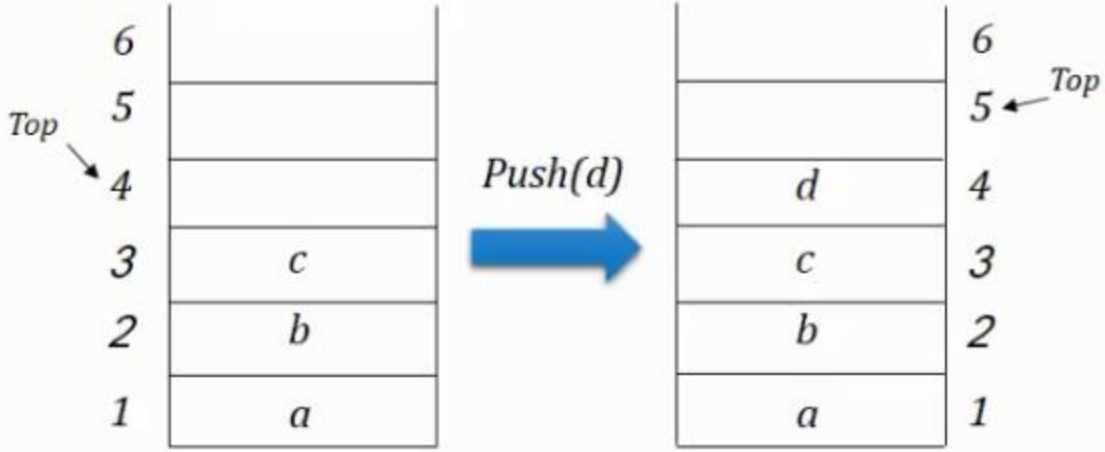
```
Void Push ( type item ){
If ( top == n+1 )
    cout<<“Stack is FULL”;
Else{
    stack[top] = item;
    top+=1;}
}
```

الگوریتم Push

وضعیت Top: اشاره به آخرین خانه پر



وضعیت Top: اشاره به اولین خانه خالی



الگوریتم Pop

وضعیت Top: اشاره به آخرین خانه پر

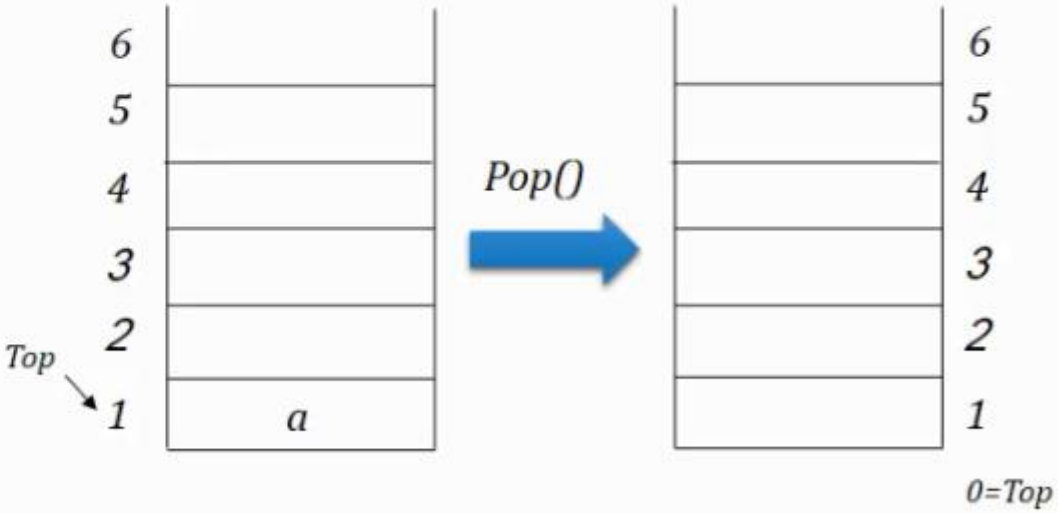
```
type Pop ( ){
    type item;
    If ( top == 0 ){
        cout<<“Stack is EMPTY”;
        return 0;}
    Else{
        item = stack[top];
        top-=1;
    return item;}
}
```

وضعیت Top: اشاره به اولین خانه خالی

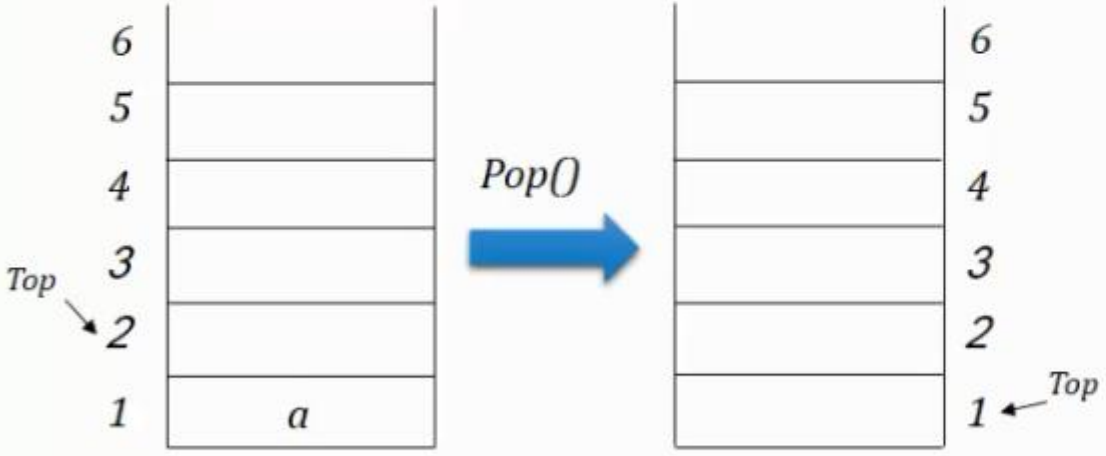
```
type Pop ( ){
    type item;
    If ( top == 1 ){
        cout<<“Stack is EMPTY”;
        return 0;}
    Else{
        top-=1;
        item = stack[top];
    return item;}
}
```

الگوریتم Pop

وضعیت Top: اشاره به آخرین خانه پر



وضعیت Top: اشاره به اولین خانه خالی



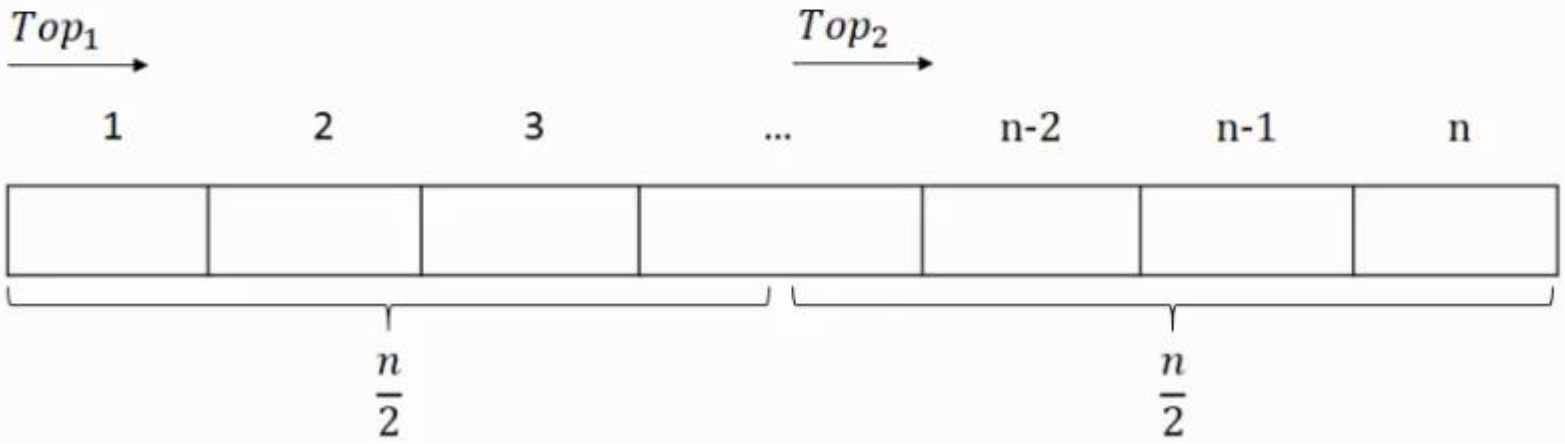
پشته دوگانه

در صورتی که بخواهیم ۲ پشته را در یک آرایه قرار دهیم، به ساختار داده حاصل شده پشته دوگانه گویند.

وضعیت های مختلفی برای مدیریت پشته ها وجود دارد:

الف) Top_1 : از ابتدا تا وسط حرکت کند.

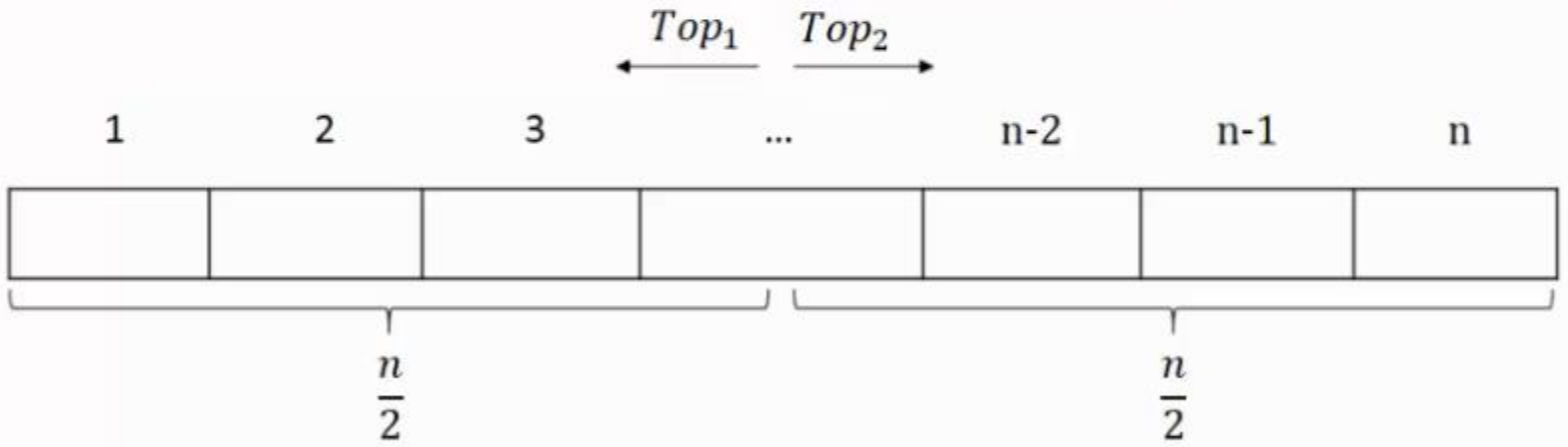
Top_2 : از وسط تا انتها حرکت کند.



پشته دوگانه

ب) Top_1 : از وسط تا ابتدا حرکت کند.

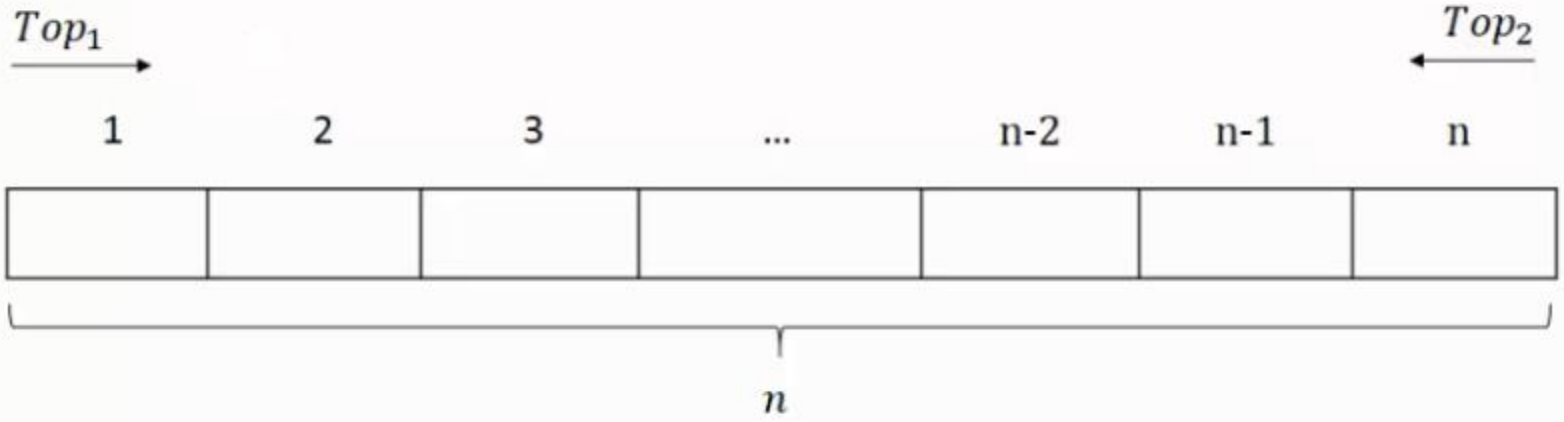
Top_2 : از وسط تا انتها حرکت کند.



پشته دوگانه

ج) Top_1 : از ابتدا تا Top_2 حرکت کند.

Top_2 : از انتها تا Top_1 حرکت کند.

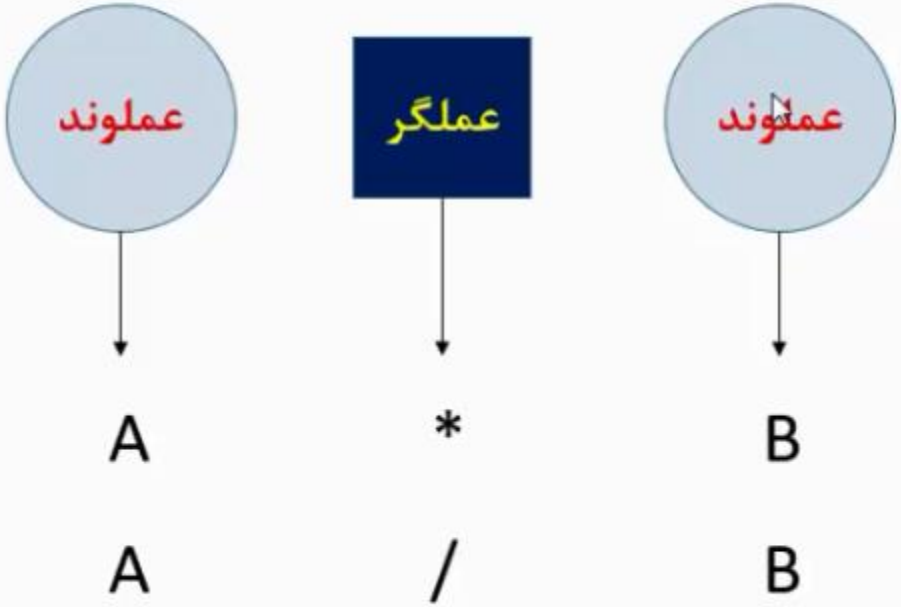


وضعیت Top_1 و Top_2	شرط پر بودن پشته ها
Top_1 و Top_2 به آخرین خانه پر اشاره کنند.	$Top_1=Top_2-1$ یا $Top_2=Top_1+1$
Top_1 و Top_2 به اولین خانه خالی اشاره کنند.	$Top_1=Top_2+1$ یا $Top_2=Top_1-1$

عبارات میانوندی–پسوندی–پیشوندی

عبارت میانوندی(infix):

به عبارت عمومی محاسباتی که عملگر دودویی بین عملوند هایش قرار می گیرد، عبارت میانوندی گویند.

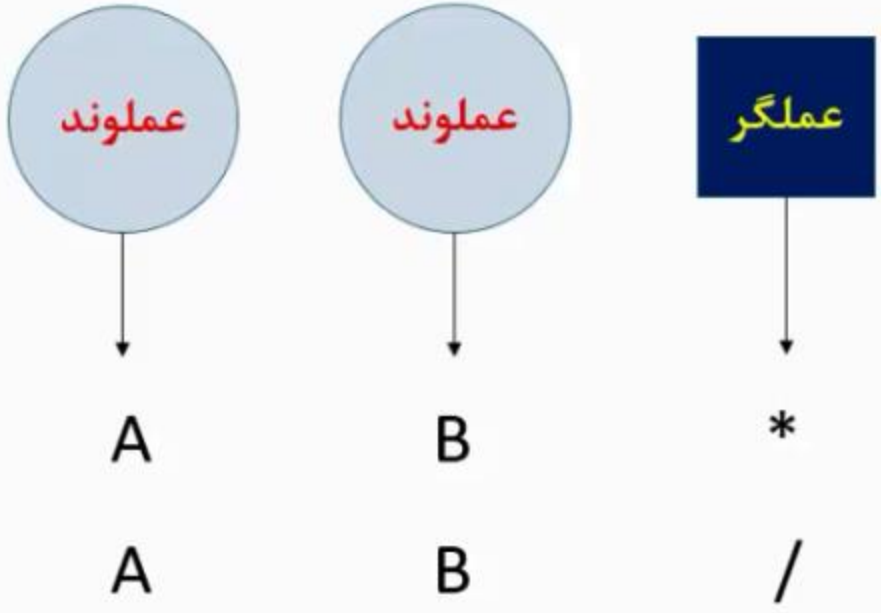


★ در عبارات میانوندی اولویت عملگر ها مهم بوده و از پرانتز گذاری برای تغییر اولویت عملگر ها استفاده می شود.

عبارات میانوندی–پسوندی–پیشوندی

عبارت پسوندی(postfix):

در این روش هر عملگر بعد از عملوند هایش قرار می گیرد. به این روش، روش لهستانی معکوس نیز می گویند.

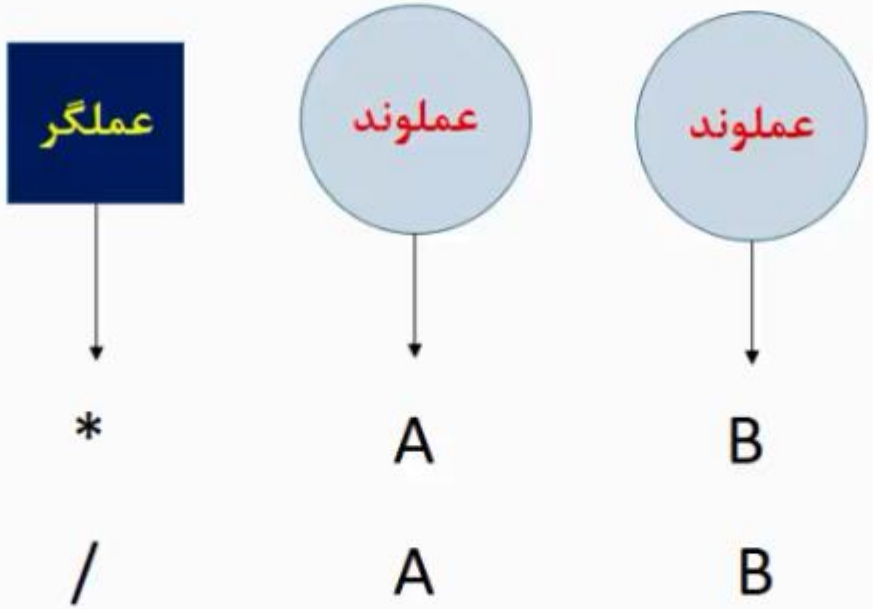


★ در عبارات پسوندی اولویت عملگر ها مهم نیست و پرانتز گذاری وجود ندارد.

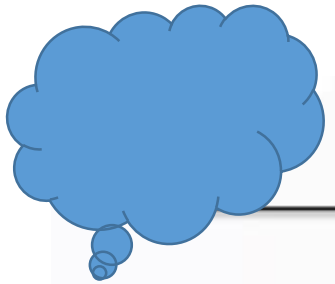
عبارات میانوندی–پسوندی–پیشوندی

عبارت پیشوندی(prefix):

در این روش هر عملگر قبل از عملوند هایش قرار می گیرد. به این روش، روش لهستانی نیز می گویند.



★ در عبارات پیشوندی اولویت عملگر ها مهم نیست و پرانتز گذاری وجود ندارد.



الگوریتم محاسبه مقدار عبارت پسوندی

۱) رشته را از چپ به راست پیمایش کنید.

۲) اگر عملوند دیدید در پشته Push کنید.

۳) اگر عملگر دیدید دو مقدار از بالای پشته Pop کنید.

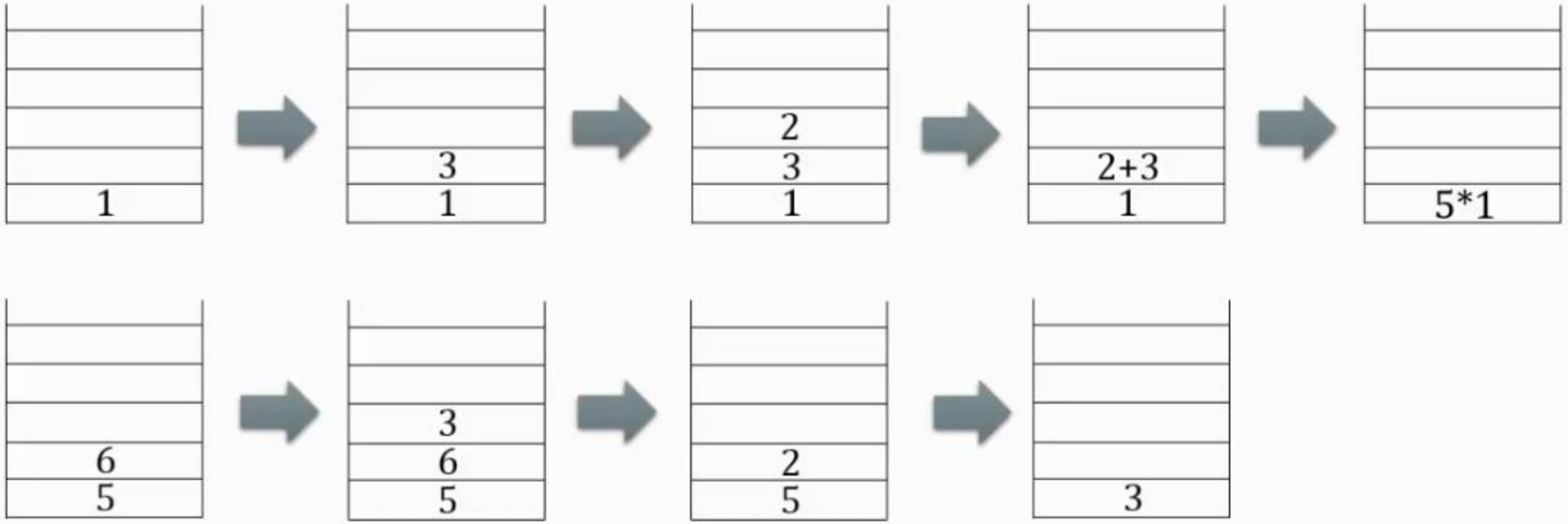
۴) عملگر دیده شده را روی آن ها اعمال کنید.

۵) حاصل را مجددا Push کنید.

الگوریتم محاسبه مقدار عبارت پسوندی

مثال: اگر a=1 و b=3 و c=2 و d=6 و e=3 باشد، حاصل عبارت پسوندی زیر را بیابید.

abc+*de/-



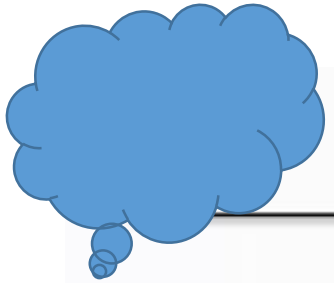
تبدیل عبارات میانوندی به پسوندی و پیشوندی

روش ها

۱) با پرانتز گذاری

۲) با پشته

۳) با پیمایش درخت عبارت محاسباتی



روش پرانتز گذاری

تبدیل infix به prefix:

۱) عبارت را بر اساس اولویت عملگرها پرانتز گذاری می کنیم. (تغییری در پرانتز گذاری اولیه نمی دهیم)

۲) عملگر مربوط به هر پرانتز را به قبل از پرانتز باز آن انتقال می دهیم.

۳) عبارت را از چپ به راست در خروجی می نویسیم.

روش پرانتز گذاری

اولویت عملگر ها

اولویت	نام عملگر	توضیحات
۱	( )	
۲	- (منفی)، + (مثبت)	
۳	^ (توان)	اولویت توان ها متوالی از راست به چپ
۴	* (ضرب)، / (تقسیم)	هم اولویت، اولویت از چپ به راست
۵	+ (جمع)، - (تفریق)	هم اولویت، اولویت از چپ به راست

مثال:

a * (b + c) – g / d ---> ((a * (b + c)) - (g/d))

---> -(*(a +(bc)) /(gd)) ---> - * a + bc / gd

روش پرانتز گذاری

مثال:

a * b + c ^ d ^ e – f ---> (((a * b) + ((c ^ (d ^ e))) – f)

---> -(+(*(ab)(^(c^(de)))f) ---> - +*ab^^cdef

مثال:

a * b ^(c + d ^-e * f) / g – h * k ---> (((a*(b^(c+((d ^(-e))*f))))/g)-(h*k))

---> -(/(*(a^(b+(c*(^(d-(e))f)))) g)*(h*k))

---> -/*a^b+c*^d-ef g*hk

روش پرانتز گذاری

تبدیل infix به postfix:

۱) عبارت را بر اساس اولویت عملگرها پرانتز گذاری می کنیم. (تغییری در پرانتز گذاری اولیه نمی دهیم)

۲) عملگر مربوط به هر پرانتز را به بعد از پرانتز بسته آن انتقال می دهیم.

۳) عبارت را از چپ به راست در خروجی می نویسیم.

مثال:

a * (b + c) – g / d ---> ((a * (b + c)) - (g/d))

---> ((a (b c)+)* (gd)/)- ---> abc+*gd/-

روش پرانتز گذاری

مثال:

a * b + c ^ d ^ e – f ---> (((a * b) + ((c ^ (d ^ e))) – f)

---> (((ab)*((c (de)^)^)+f)- ---> ab*cd^e^+f-

مثال:

a * b ^(c + d ^-e * f) / g – h * k ---> (((a*(b^(c+((d ^(-e))*f))))/g)-(h*k))

---> (((a(b(c((d(e)-)^f)*)+)^)*g)/(hk)*)-

--->abcde-^f*+^*g/hk*-

تبدیل با پشته

تبدیل infix به prefix:

۱) عبارت را از راست پیمایش می کنیم.

۲) اگر به عملوند رسیدیم، آن را در خروجی می نویسیم.

۳) اگر به ")" که رسیدیم آن را در پشته قرار می دهیم.

۴) اگر به عملگر رسیدیم، در صورتی که اولویت آن عملگر از عملگر بالای پشته بیشتر یا مساوی باشد، آن را به پشته اضاف می کنیم. در غیر اینصورت آن قدر از بالای پشته عملگر خارج می کنیم و در خروجی می نویسیم تا یا پشته خالی شود و یا به عملگری برسیم که اولویت آن از عملگر مورد نظر کمتر باشد.

تبدیل با پشته

تبدیل infix به prefix:

۵) اگر بالای پشته "(" بود هر عملگری به راحتی روی آن قرار می گیرد.

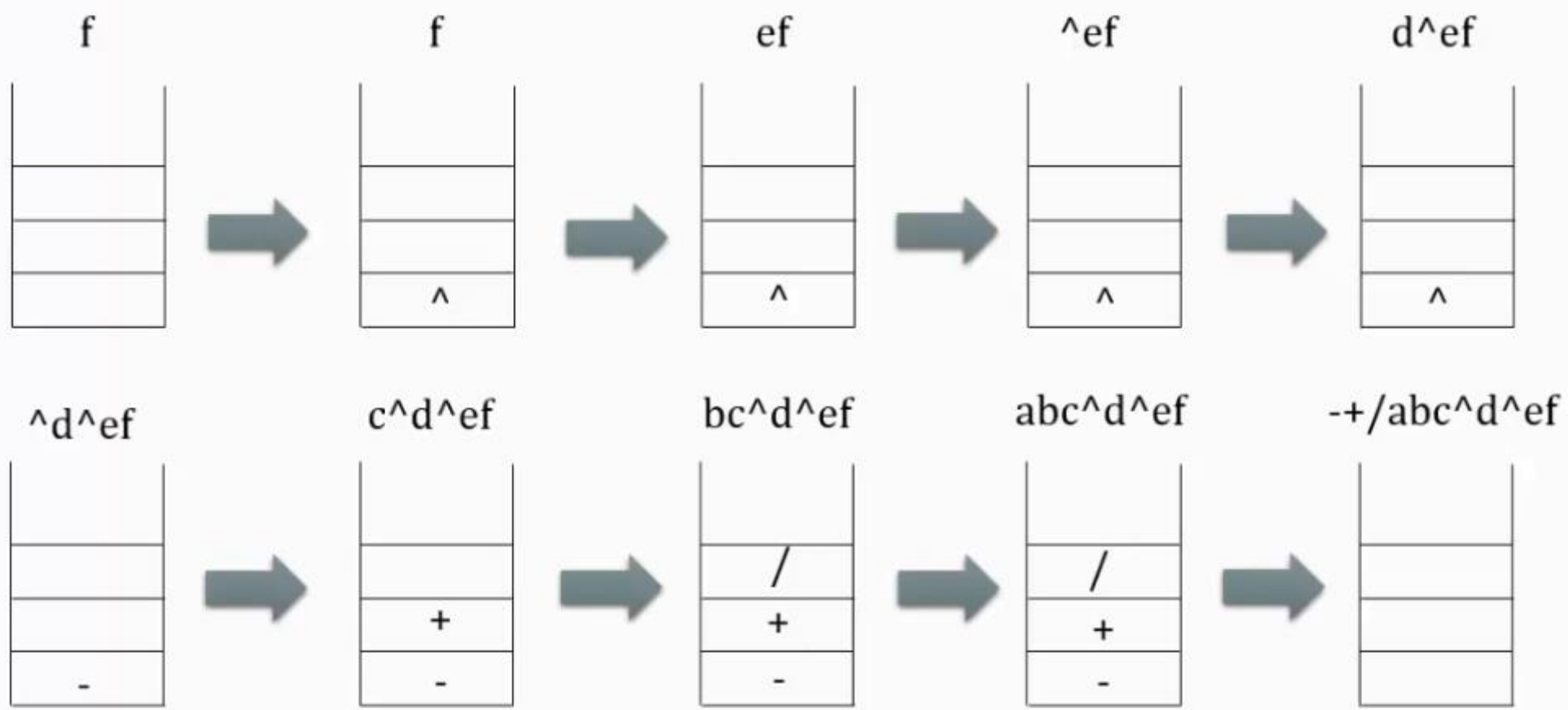
۶) به هر ")" که رسیدیم آن قدر از پشته عملگر خارج می کنیم و در خروجی می نویسیم تا به "(" برسیم. در این وضعیت "(" با ")" خنثی شده و "(" از پشته حذف می شود.

۷) زمانی که به انتهای عبارت رسیدیم در صورتی که پشته خالی نباشد، عملگر ها را به ترتیب از پشته خارج کرده و در خروجی می نویسیم.

★با توجه به مورد"۴" عملگر - و + می توانند روی – یا + قرار گیرند، همچنین * و / می توانند روی * یا / قرار گیرند. اما توان روی توان نمی تواند قرار گیرد.

تبدیل با پشته

مثال: عبارت میانوندی a/b+c-d^e^f را با استفاده از پشته، به عبارت پیشوندی تبدیل کنید.



تبدیل با پشته

تبدیل infix به postfix:

۱) عبارت را از چپ پیمایش می کنیم.

۲) اگر به عملوند رسیدیم، آن را در خروجی می نویسیم.

۳) اگر به ")" که رسیدیم آن را در پشته قرار می دهیم.

۴) اگر به عملگر رسیدیم، در صورتی که اولویت آن عملگر از عملگر بالای پشته بیشتر باشد، آن را به پشته اضاف می کنیم. در غیر اینصورت آن قدر از بالای پشته عملگر خارج می کنیم و در خروجی می نویسیم تا یا پشته خالی شود و یا به عملگری برسیم که اولویت آن از عملگر مورد نظر کمتر باشد.

تبدیل با پشته

تبدیل infix به postfix:

۵) اگر بالای پشته ")" بود هر عملگری به راحتی روی آن قرار می گیرد.

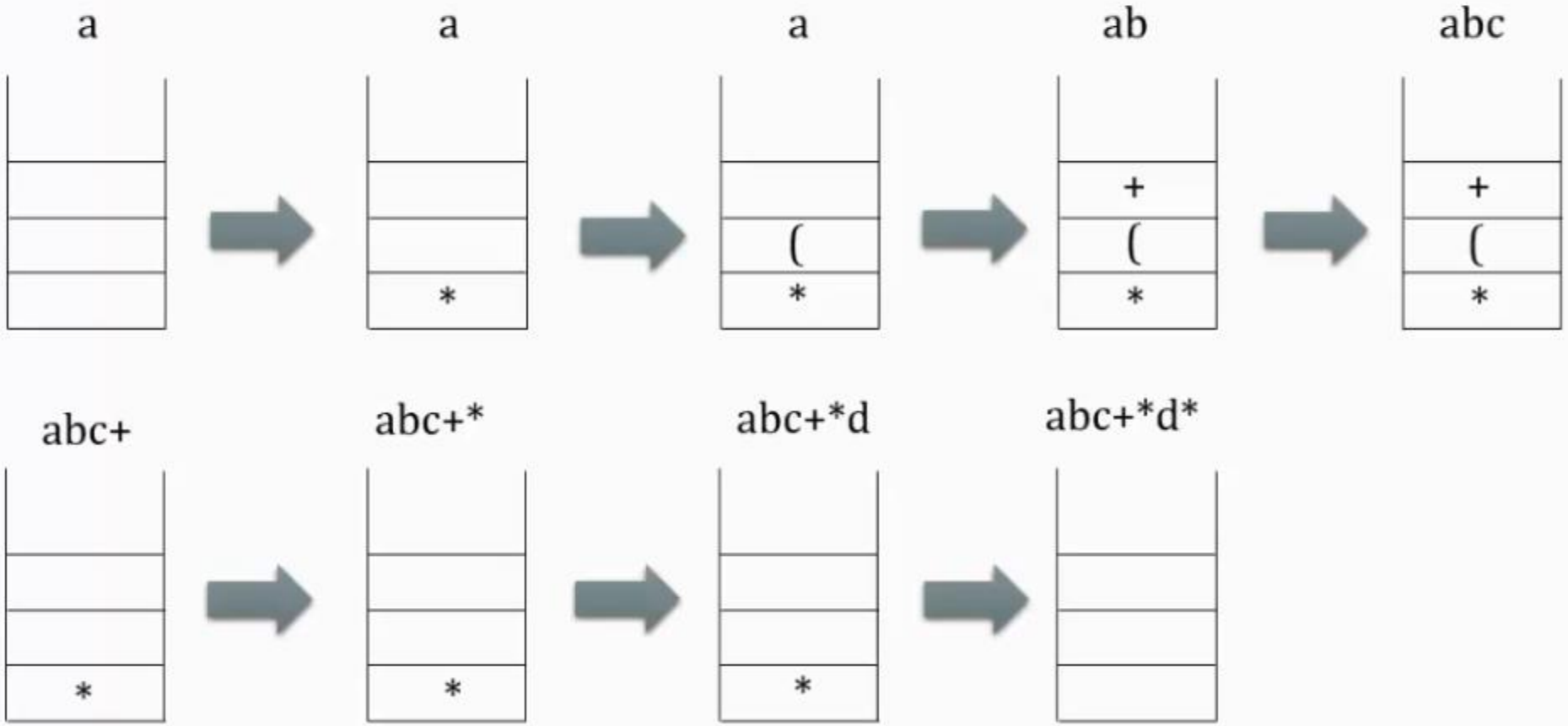
۶) به هر "(" که رسیدیم آن قدر از پشته عملگر خارج می کنیم و در خروجی می نویسیم تا به ")" برسیم. در این وضعیت "(" با ")" خنثی شده و ")" از پشته حذف می شود.

۷) زمانی که به انتهای عبارت رسیدیم در صورتی که پشته خالی نباشد، عملگر ها را به ترتیب از پشته خارج کرده و در خروجی می نویسیم.

★با توجه به مورد"۴" عملگر - و + نمی توانند روی – یا + قرار گیرند، همچنین * و / هم نمی توانند روی * یا / قرار گیرند. اما توان روی توان می تواند قرار گیرد.

تبدیل با پشته

مثال: عبارت میانوندی a*(b+c)*d را با استفاده از پشته، به عبارت پسوندی تبدیل کنید.



روش پرانتز گذاری

تبدیل prefix به infix :

۱) عبارت را از راست به چپ پیمایش می کنیم.

۲) با رسیدن به هر عملگر، دو عملوند سمت راست آن را به همراه عملگر، داخل یک پرانتز قرار می دهیم.

۳) عبارت را از چپ به راست پیمایش می کنیم و هر عملگر را بین عملوند هایش قرار می دهیم.

مثال:

+-a/bcd ---> +-a(/bc)d ---> +(-a(/bc))d ---> (+(-a(/bc))d)

---> ((a-(b/c))+d) ---> a-b/c+d

روش پرانتز گذاری

مثال:

+-*^abcd/e/f+gh ---> +-*^abcd/e/f(+gh) ---> +-*^abcd/e(/f(+gh))

---> +-*^abcd(/e(/f(+gh))) ---> +-*(^ab)cd(/e(/f(+gh)))

---> +-(*(^ab)c)d(/e(/f(+gh))) ---> +(-(*(^ab)c)d) (/e(/f(+gh)))

---> (+(-(*(^ab)c)d) (/e(/f(+gh)))) ---> ((((a^b)*c)-d)+(e/(f/(g+h))))

---> a^b*c-d+e/f/g+h

روش پرانتز گذاری

تبدیل postfix به infix :

۱) عبارت را از چپ به راست پیمایش می کنیم.

۲) با رسیدن به هر عملگر، دو عملوند سمت چپ آن را به همراه عملگر، داخل یک پرانتز قرار می دهیم.

۳) عبارت را از چپ به راست به ترتیب شامل عملوندها و عملگر های بین عملوند ها ارزیابی میکنیم.

مثال:

abc*d/+ ---> a(bc*)d/+ ---> a((bc*)d/)+ ---> (a((bc*)d/)+)

---> (a+((b*c)/d)) ---> a+b*c/d

روش پرانتز گذاری

مثال:

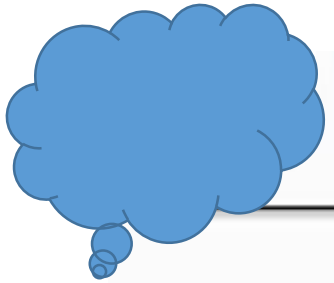
ab^c*d- efgh+//+ ---> (ab^)c*d- efgh+//+ ---> ((ab^)c*)d- efgh+//+

---> (((ab^)c*)d-) efgh+//+ ---> (((ab^)c*)d-) ef(gh+)//+

---> (((ab^)c*)d-) e(f(gh+)/)/+ ---> (((ab^)c*)d-) (e(f(gh+)/)/)+

---> ((((ab^)c*)d-) (e(f(gh+)/)/)+) ---> ((((a^b)*c)-d)+(e/(f/(g+h))))

---> a^b*c-d+e/f/g+h



تبدیل با پشته

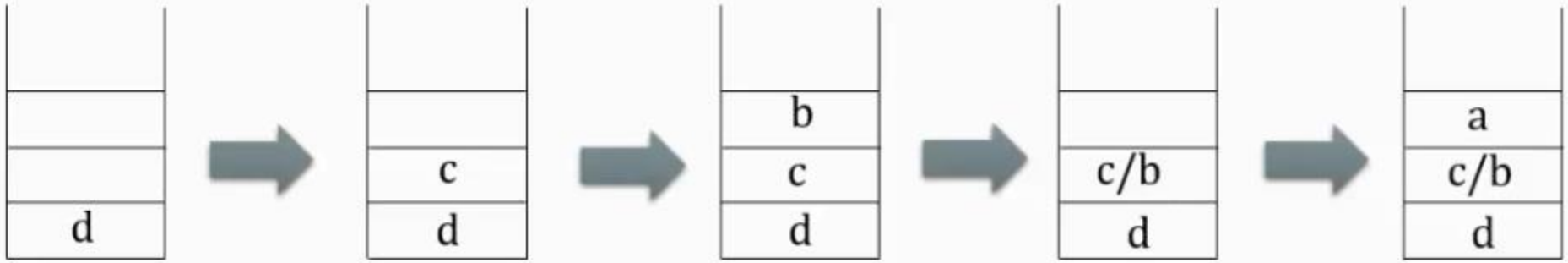
تبدیل prefix به infix :

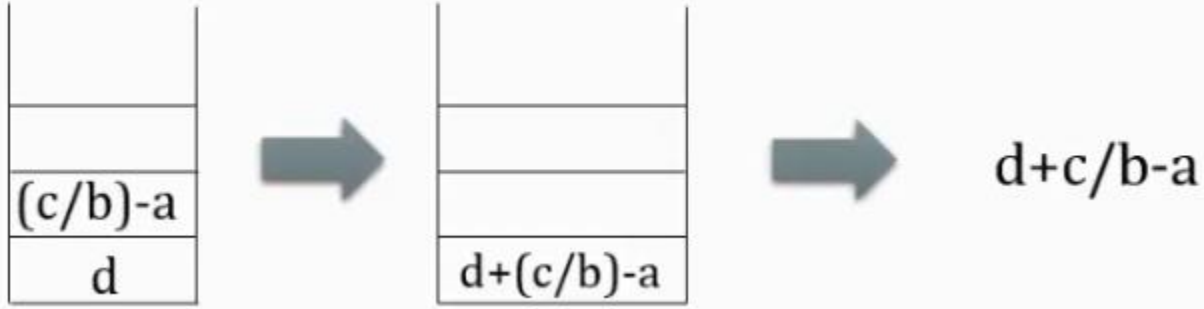
۱) عبارت را از راست به چپ پیمایش می کنیم.

۲) عملوند ها را در پشته قرار می دهیم و با رسیدن به هر عملگر دو عملوند خارج کرده و بعد از محاسبه دوباره حاصل عبارت را داخل پشته قرار می دهیم.

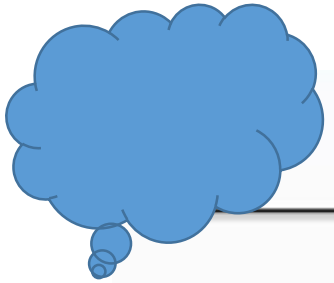
تبدیل با پشته

مثال: عبارت پیشوندی +-a/bcd را با استفاده از پشته، به عبارت میانوندی تبدیل کنید.





d+c/b-a



تبدیل با پشته

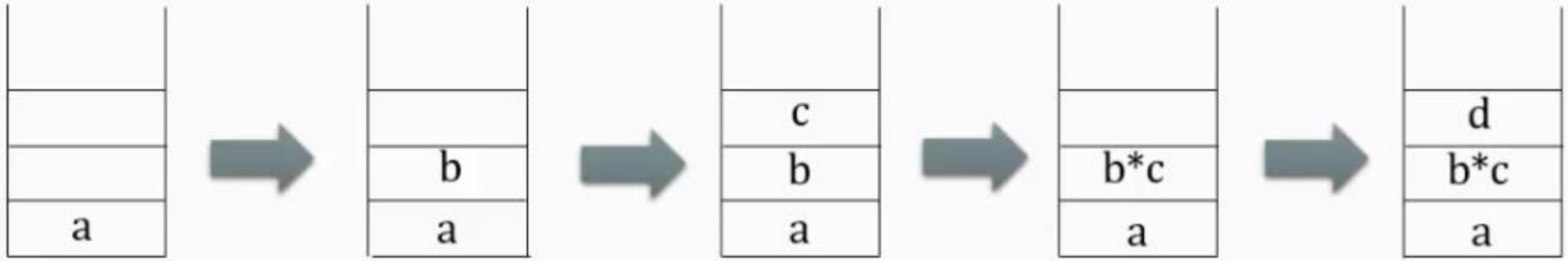
تبدیل postfix به infix :

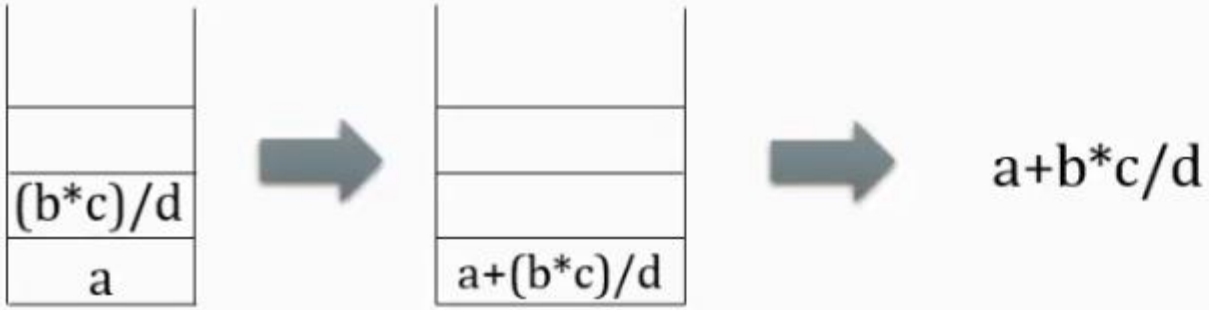
۱) عبارت را از چپ به راست پیمایش می کنیم.

۲) عملوند ها را در پشته قرار می دهیم و با رسیدن به هر عملگر دو عملوند خارج کرده و بعد از محاسبه دوباره حاصل عبارت را داخل پشته قرار می دهیم.

تبدیل با پشته

مثال: عبارت پیشوندی abc*d/+ را با استفاده از پشته، به عبارت میانوندی تبدیل کنید.





a+b*c/d

تبدیل prefix به postfix و برعکس

تبدیل prefix به postfix :

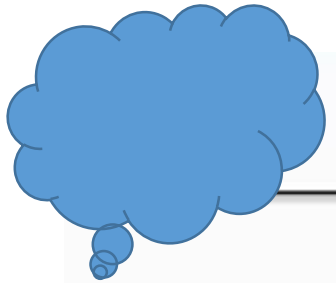
مشابه الگوریتم تبدیل prefix به infix با روش پرانتز گذاری عمل می کنیم، فقط در انتها به جای قرار دادن عملگر بین عملوندها، عملگر را روی پرانتز بسته می گذاریم.

در نهایت به فرم postfix خواهیم رسید.

تبدیل postfix به prefix :

مشابه الگوریتم تبدیل postfix به infix با روش پرانتز گذاری عمل می کنیم، فقط در انتها به جای قرار دادن عملگر بین عملوندها، عملگر را روی پرانتز باز می گذاریم.

در نهایت به فرم prefix خواهیم رسید.



تبدیل prefix به postfix و برعکس

مثال:

*/*xy+cd-ab ---> */*xy+cd(-ab) ---> */*xy(+cd)(-ab)

---> */(*xy)(+cd)(-ab) ---> *(/(*xy)(+cd)) (-ab)

---> (*(/(*xy)(+cd)) (-ab)) ---> (((xy*)(cd+)/) (ab-)*)

---> xy*cd+/ab-*