Data Structure

Dr. Rastgoo

فصل ۴:

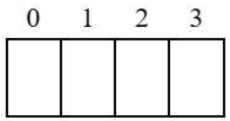
صف و پشته

صف

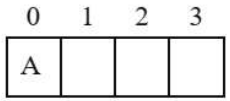
صف (queue)، ساختمان داده ای است که عمل حذف از ابتدای آن و درج به انتهای آن انجام می‌شود. صف را لیست (First In First Out) FIFO می نامند، زیرا اولین عنصر وارد شده به صف، اولین عنصری است که خارج می‌شود. برای نمایش صف، از آرایه یک بعدی queue[0..n-1] و دو متغیر front و rear استفاده می‌شود. در متغیرfront یکی کمتر از مقدار محل عنصر اول صف و در متغیر rear محل آخرین عنصر صف ذخیره می شود.

مثال

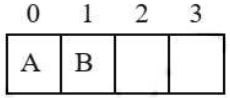
یک صف با 4 خانه که در ابتدا خالی است مفروض می‌باشد. (front = rear = -1)



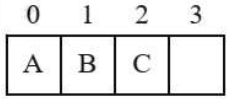
front = -1 , rear = -1



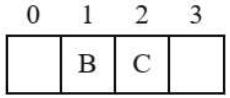
front = -1 , rear = 0 — **درج A**



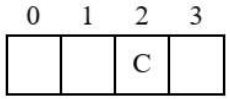
front = -1 , rear = 1 — **درج B**



front = -1 , rear = 2 — **درج C**



front = 0 , rear = 2 — **حذف A**



front = 1 , rear = 2 — **حذف B**

در صف خالی مقدار front با rear برابر است و در صف پر مقدار rear برابر n-1 می باشد.

اضافه کردن عنصر به صف

```
void  addq (int  *rear , int   item){
  if (*rear == max -1)
    {
      queue-full( );
      return;
    }
  queue[++*rear] = item;
}
```

(max : حداکثر اندازه صف می باشد.)

حذف عنصر از صف

```
int  deleteq (int  *front , int   rear){
  if (*front == rear)
       return queue-empty( );
  return
      q[++*front];
}
```

این نوع نمایش ترتیبی صف، دارای نقاط ابهامی است. با ورودی و خروج داده ها ، صف بتدریج بطرف راست تغییر مکان می دهد. به نحوی که rear برابر max-1 می شود و به نظر می رسد که صف پر است. در این حالت ، queue-full باید تمام صف را به سمت چپ شیفت دهد و اولین عنصر دوباره در موقعیت queue[0] قرار گرفته و front برابر 1- شود. rear نیز باید تنظیم شود. ولی شیفت عملی زمان گیر است.

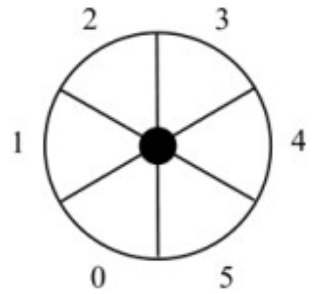
صف حلقوی

در صف ساده مشکلاتی داشتیم. اگر آرایه را حلقوی فرض کنیم، اندیس ابتدا همیشه به یک موقعیت عقب تر (در خلاف حرکت عقربه های ساعت) از اولین عنصر موجود در صف اشاره می کند. اندیس انتها(rear) به انتهای فعلی صف اشاره می کند. اگر front=rear باشد، صف خالی خواهد بود.

در یک صف حلقوی به اندازه max حداکثر max-1 عنصر می تواند، قرار گیرد. اگر از آن یک خانه نیز استفاده شود، front=rear می شود و نمی توانیم یک صف پر و خالی را از هم تشخیص دهیم.

مثال

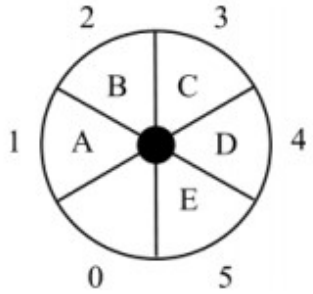
حالتهای خالی و پر در یک صف حلقوی در زیر نمایش داده شده است:



صف خالی

front = 0

rear =0



صف پر

front = 0

rear = 5

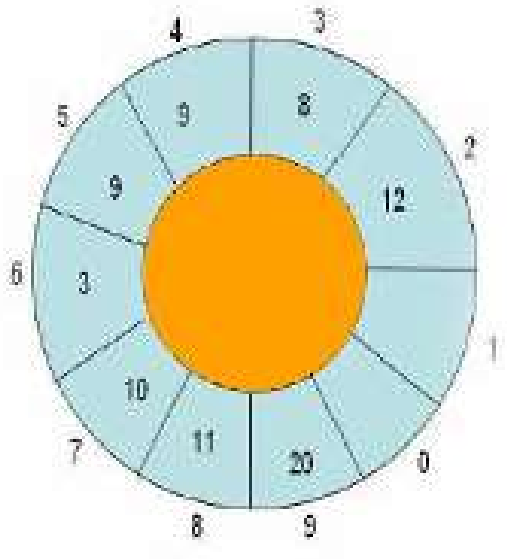
- اگر شرط front = rear در صف حلقوی برقرار باشد، آنگاه صف حلقوی خالی است.
- اگر شرط (rear+1) mod n = front در صف حلقوی برقرار باشد، آنگاه صف حلقوی پر است.
- اگر شرط (n-front+rear) mod n = n-1 در صف حلقوی برقرار باشد، آنگاه صف حلقوی پر است.
- تعداد عناصر صف حلقوی برابر (n-front+rear) mod n است که می توان به صورت زیر نیز بیان کرد:

اگر front≤rear : rear - front

و اگر front>rear : n - (front - rear)

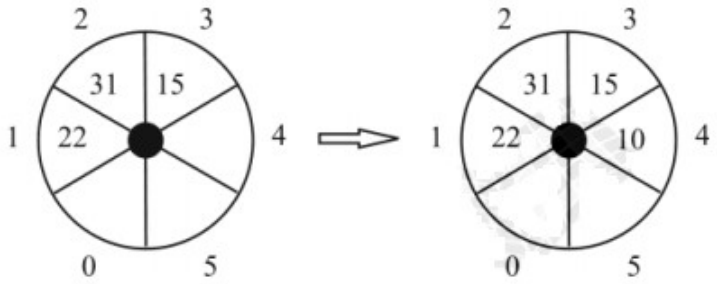
درج و حذف در صف حلقوی

برای اضافه کردن یک عنصر لازم است که rear در جهت عقربه‌های ساعت حرکت کند و اگر با front برابر شد آنگاه صف پر است و توسط تابع full مقدار rear به مقدار قبلی برگردانده می شود و پیغام خطا چاپ می شود.



مثال

اضافه کردن عدد 10 به صف حلقوی زیر:



front = 0
rear = 3

front = 0
rear = 4

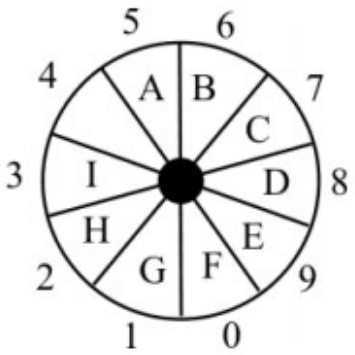
```
addq (front , *rear , item)
{
   *rear = (*rear+1) % max;
    if ( *rear == front)
        {
          queue-full(rear);
          return;
        }
   queue[*rear] = item ;
}
```

```
deleteq(*front , rear)
{
  if (*front == rear)
         return queue-empty( );
  *front=(*front+1) % max;
   return   queue[*front];
}
```

مثال

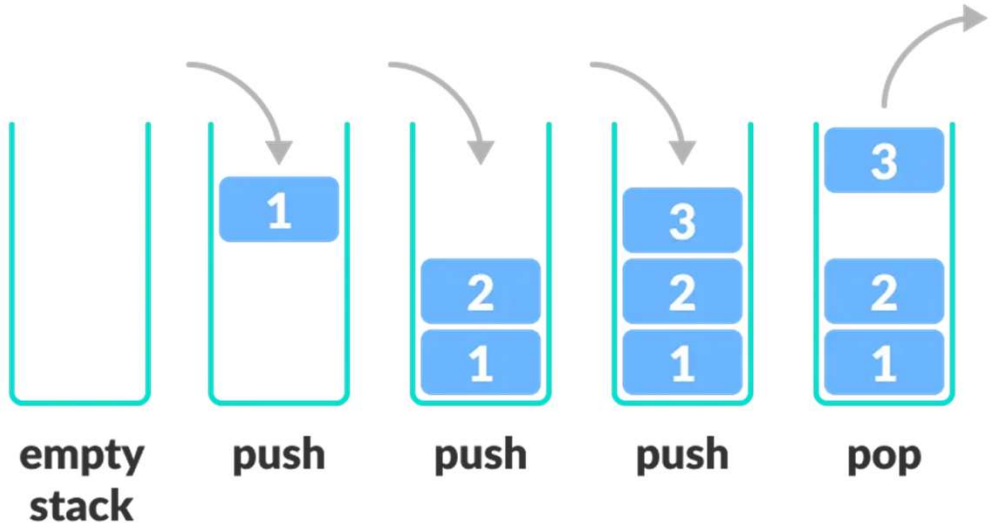
یک صف حلقوی با ۱۰ خانه و front=4 , rear=3 ، چه وضعیتی دارد؟

حل: صف پر است، چون مقدار (rear+1) % n برابر front است. شکل زیر مثالی از این حالت است:

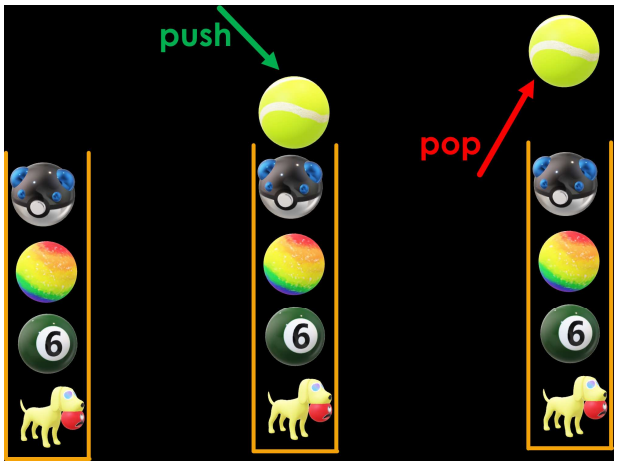


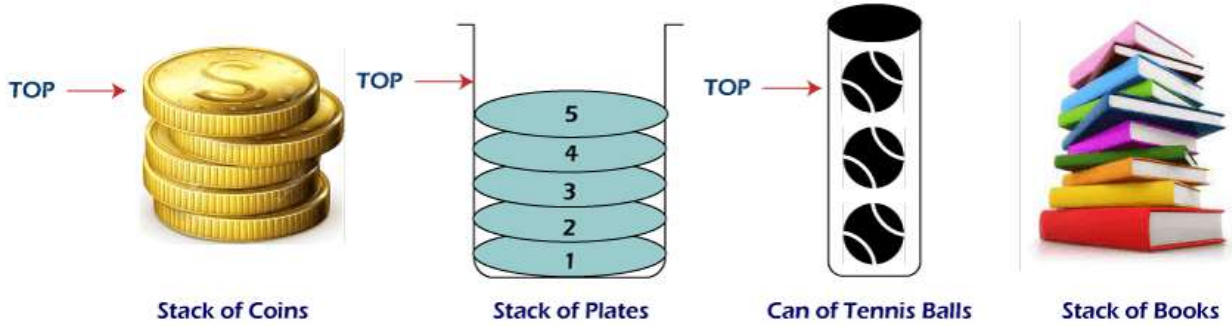
پشته (STACK)

پشته ساختمان داده ای نیمه ایستا می باشد که حذف و اضافه از بالای آن انجام می‌شود. پشته را LIFO بـه معنـی In First Last Out می‌نامند، چون آخرین عنصر وارد شده در آن، اولین عنصری است که از آن برداشته می‌شود.



ساده ترین روش پیاده سازی پشته، استفاده از یک آرایه یک بعدی به نام stack[MAX] است که MAX ، حداکثر تعداد عناصر آرایه می باشد. همچنین از یک متغیر به نام top که به عنصر بالایی پشته اشاره می کند، نیاز است. اولین یا پایین ترین عنصر پشته در stack[0] ذخیره می شود. در ابتدا به top مقدار 1- داده می شود که نشان دهنده یک پشته خالی است.

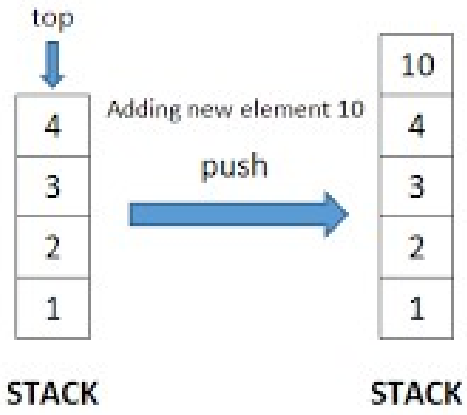




عملیات درج در پشته

```
void  push (int  item , int *top){
   if (*top >= max-1)
     {
      stack-full( );
      return;
     }
   stack[++*top] = item;
}
```

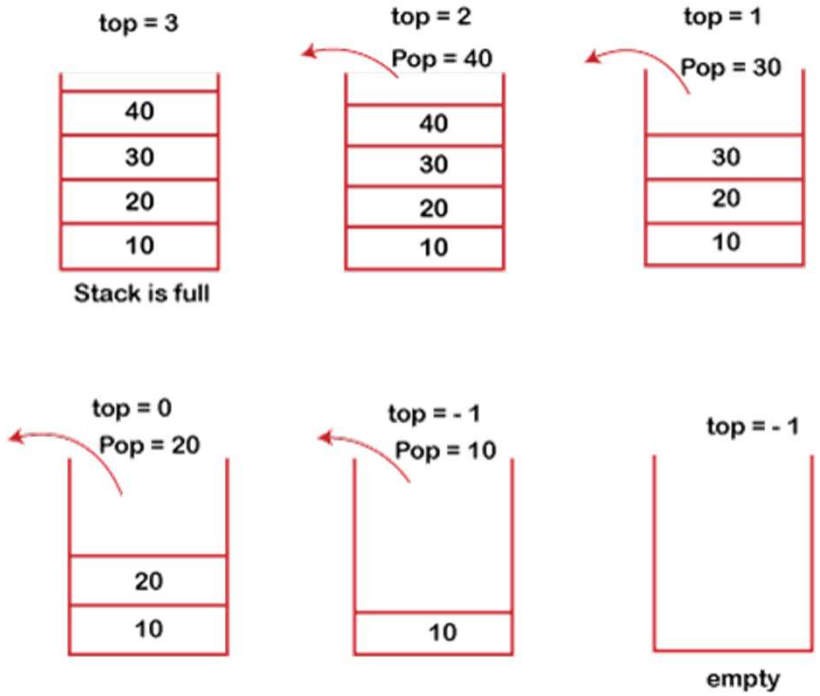
در این تابع اگر پشته پر نباشد، top را افزایش داده و item در آرایه stack اضافه می شود.

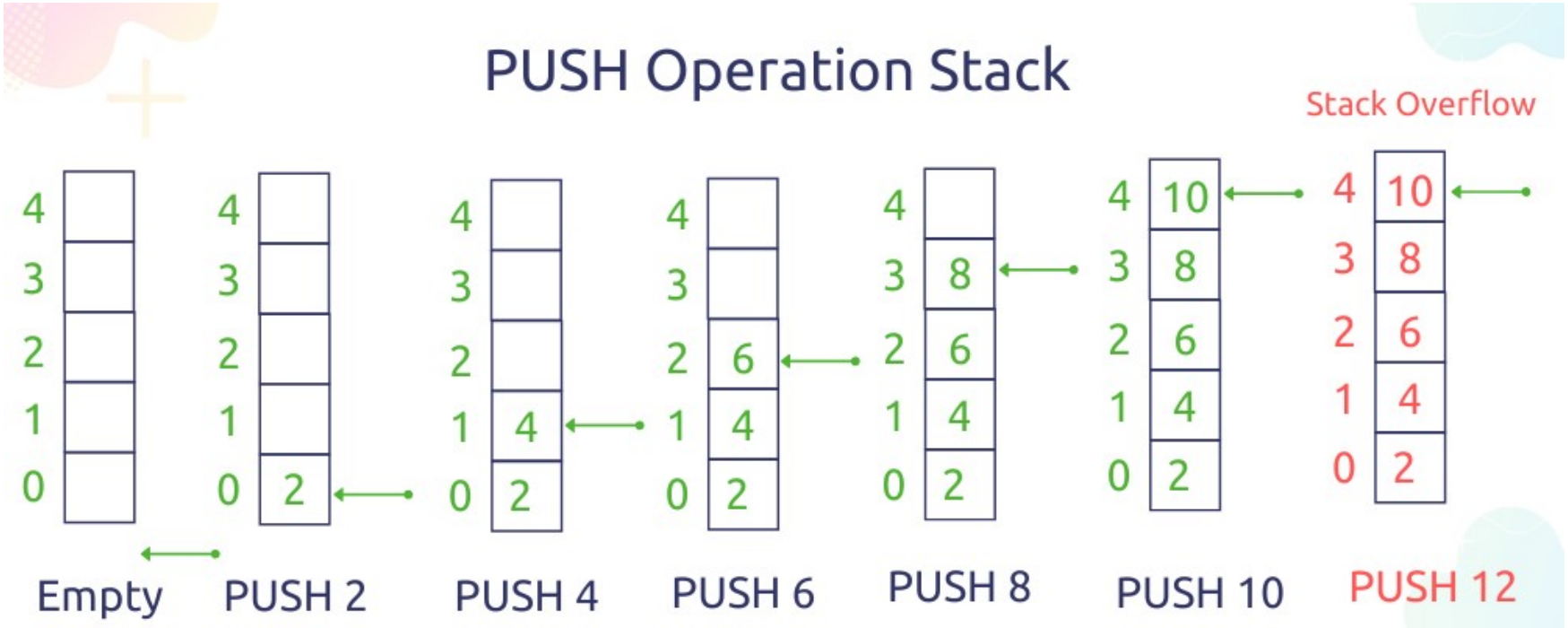


عملیات حذف از پشته

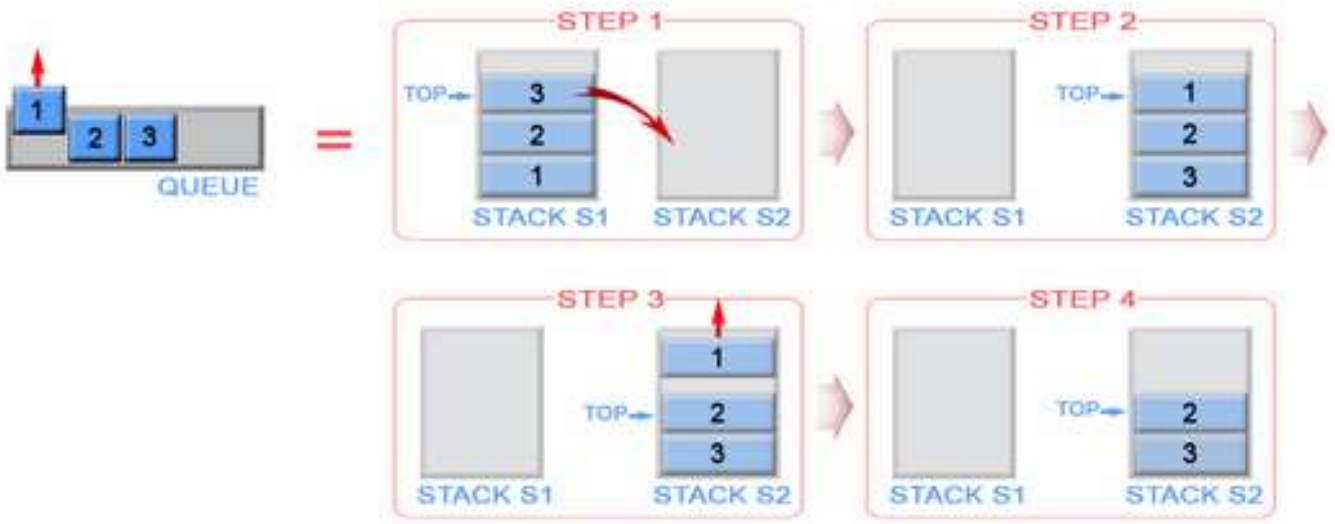
```
int  pop(int  *top){
    if (*top == -1)
       return stack-empty( );
    return  stack[(*top)--];
}
```

تابع اگر پشته خالی نباشد، عنصر بالای پشته برگردانده شده و مقدار top یک واحد کاهش می یابد.

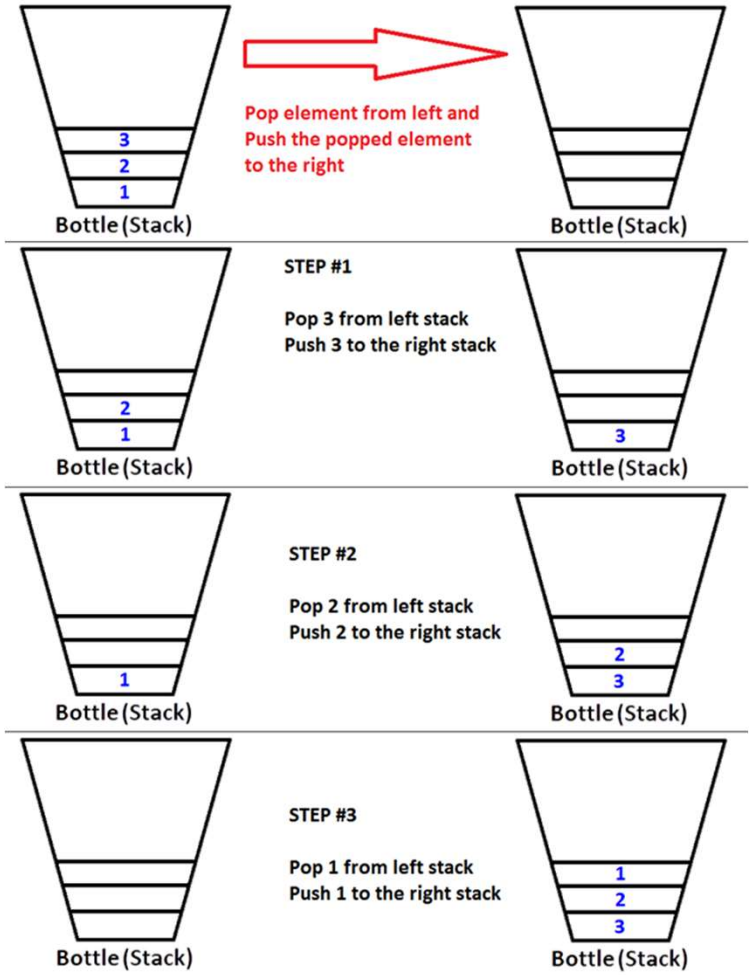



PUSH Operation Stack
Stack Overflow
4
3
2
1
0
2
6
4
8
10
Empty
PUSH 2
PUSH 4
PUSH 6
PUSH 8
PUSH 10
PUSH 12

برای معکوس کردن محتویات صف از یک پشته کمک می گیریم. ابتدا تمامی عناصر صف را حذف کرده و به پشته اضافه می کنیم، سپس تمامی عناصر پشته را حذف کرده و به صف بر می گردانیم. برای معکوس کردن ترتیب عناصر یک پشته از یک صف اضافی کمک می گیریم.



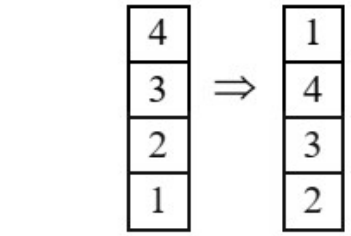
Popping element "1" from the queue



استفاده از دو پشته

مثال

اگر بخواهیم عنصر پایین پشته زیر را بر روی عناصر پشته قرار دهیم، به چند push نیاز داریم؟



پاسخ:

مراحل کار به صورت زیر است:

۱- سه عدد بالای پشته را pop کرده و در یک پشته کمکی push می کنیم.

۲- عدد 1 را pop کرده و در یک متغیر کمکی ذخیره می کنیم.

۳- سه عدد موجود در پشته کمکی را pop کرده و در پشته اصلی push می کنیم.

۴- مقدار موجود در متغیر کمکی (یعنی 1) را در پشته اصلی push می کنیم.

بنابراین به هفت عدد push نیاز است.

مثال

روی هم چند push وpop، برای انتقال عناصر از پشته ۱، به همان ترتیب، به پشته ۲ نیاز است؟

پاسخ:

ابتدا با n تا pop عناصر را از پشته اول برداشته و با n تا push در پشته کمکی قرار می دهیم. سپس عناصر از پشته کمکی را با n تا pop برداشته و با n تا push آنها را در پشته دوم درج می کنیم. بنابراین در کل به 4n عمل push وpop نیاز داریم.

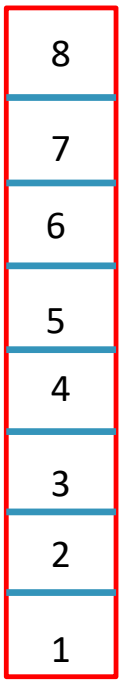
تعداد خروجی های مجاز از یک پشته n تایی برابر است با : $\frac{\binom{2n}{n}}{(n+1)}$

مثال

در ورودی یک پشته اعداد 1 تا n به ترتیب قرار دارند(۱ در ابتدای ورودی است). عمل pop و push به صورت زیر تعریف شده اند.

Push : اولین عدد ورودی را برداشته و در بالای پشته قرار می دهد.

Pop : عدد بالای پشته را برداشته و در انتهای دنباله خروجی می نویسد.



با ترکیب مناسبی از n عدد push وn عدد pop می تـوان جایگشـتی از اعـداد 1 تـا n را در خروجـی تولیـد کـرد کـه بـه آن جایگشت قابل قبول می گوییم. آیا برای n=8 ، جایگشت $< 4,3,7,8,6,2,5,1 >$قابل قبول است؟

پاسخ: خیر- چون نمی توان 2 را قبل از 5 ، pop کرد.(5 بالای 2 در پشته قرار دارد.)

راه تشخیص سریع: برای هر عدد x ، اعداد کوچکتر از x که بعد از آن قرار دارند، باید یک دنباله نزولی باشـند. در دنبالـه داده شده، اعداد بعد از 6 که $< 2,5,1 >$هستند، یک دنباله نزولی نمی باشند.

مثال

در صورتی که اعداد ۱و۲و۳ به ترتیب از راست به چپ وارد یک پشته شوند، چند حالت خروجی می تواند ایجاد شود؟

حل: با قرار دادن n=3 در رابطه $\frac{\binom{2n}{n}}{(n+1)}$ جواب برابر 5 خواهد شد.

این پنج حالت به صورت زیر می باشد:

123 (ابتدا ۱ را در پشته قرار داده و سپس آن را خارج کرده و همین کار را برای ۲و۳ نیز انجام می دهیم.)

132 (ابتدا ۱ را در پشته قرار داده و سپس آن را بر می داریم. بعد عدد ۲ و۳ را در پشته قرار داده و سپس هر دو را خارج می کنیم.)

213 (عدد ۱و۲ را در پشته قرار داده و سپس هر دو را خارج کرده و ۳ را در پشته قرار داده و خارج می کنیم.)

231 (عدد ۱و۲ را در پشته قرار داده ، عدد ۲ را خارج کرده و عدد ۳ را قرار داده و عدد ۳و۱ را خارج می کنیم.)

321 (عدد ۱و۲و۳ را وارد کرده و سپس هر سه عدد را خارج می کنیم.)

تذکر: توجه کنید که خروجی 312 ممکن نمی باشد. چون ابتدا 1 و2و3را در پشته قرار داده و سپس 3 را خارج کرده و بعد از آن نمی توان 1 را خارج کرد چون قبل از آن باید عدد 2 را خارج کرد.

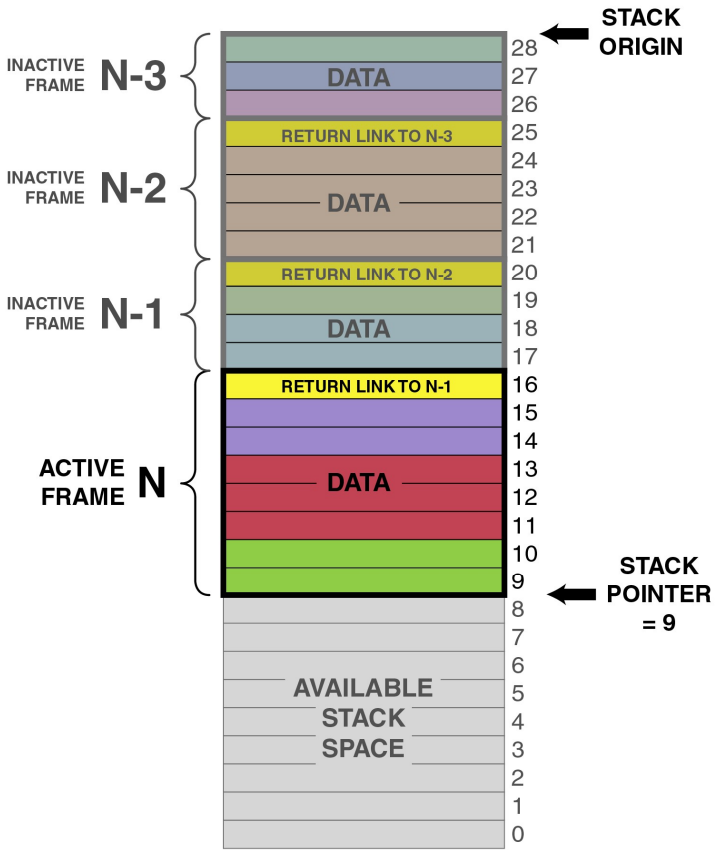
کاربردهای پشته

از کاربردهای پشته می توان موارد زیر را نام برد:

۱- ارزشیابی عبارات

۲- زیر برنامه های بازگشتی

۳- نگهداری آدرس برگشت زیر برنامه ها



ارزشیابی عبارات

یکی از کاربردهای پشته، ارزشیابی عبارات می باشد. یک عبارت از عملوندها و عملگرها ساخته شده که می‌تواند به سه شکل نمایش داده شود:

۱) prefix (پیشوندی) ۲) infix (میانوندی) ۳) postfix (پسوندی)

در فرم infix، عملگرها بین عملوندها قرار می‌گیرند. در فرم postfix، عملگرها بعد از عملوندها قرار می‌گیرند و در فرم prefix، عملگرها قبل از عملوندها قرار می‌گیرند. به طور نمونه عبارت A*B/C میانوندی، عبارت AB*C/ پسوندی و عبارت /*ABC پیشوندی می باشد.

تبدیل فرم ها به یکدیگر

حالتهای ممکن تبدیل فرم به صورت زیر می باشد:

۱- infix به postfix

۲- infix به prefix

۳- postfix به infix

۴- prefix به infix

۵- postfix به prefix

۶- prefix به postfix

تبدیل از فرم infix به postfix

ابتدا عبارت را بر حسب اولویت عملگرهایش در نظر گرفته و سپس هر عملگر را بعد از عملوندهایش می‌نویسیم. به طور نمونه عبارت A+B را به صورت +AB نمایش می دهیم.

مثال

عبارت میانوندی A/B-C+D*E را به فرم پسوندی تبدیل کنید.(اولویت: / ، * ، - ، و +)

A/B–C+D*E

تقسیم : **AB/** - C + D*E

ضرب : **AB/** - C + **DE***

منها : **AB/C-** + **DE***

جمع : **AB/C-DE*+**

تذکر: تقسیم و ضرب دارای اولویت یکسانی می باشند و هر کدام که از چپ به راست، زودتر استفاده شود، تقدم بیشتری خواهد داشت.(برای جمع و تفریق هم چنین است)

مثال

عبارت ((A+B)*D)^(E-F) را به فرم پسوندی تبدیل کنید.

(**AB**+ *D) ^ (E-F)

⇒ **AB+D*** ^ (E-F)

⇒ **AB+D*^ EF-**

⇒ **AB+D*EF-^**

تبدیل از فرم infix به prefix

در این تبدیل عملگرها را با توجه به اولویت آنها به سمت چپ منتقل می کنیم. به طور نمونه عبارت میانوندی A+B را به فرم AB+ نمایش می دهیم.

مثال

عبارت x+y*z-w را به فرم پیشوندی تبدیل نمایید.

x+ ***yz** -w $\Rightarrow$ **+x*yz** - w $\Rightarrow$ **-+x*yzw**

مثال

عبارت a+(b-c*d)^e-f^g^(h/ **i*k)** را به فرم پیشوندی و پسوندی تبدیل نمایید.

پاسخ: عملگر توان سمت راست،زودتر اجرا می شود.

-+a^-b*cde^f^g*/hik و abcd*-e^+fghi/k*^^-

تبدیل از فرم postfix به infix

مثال

تبدیل عبارت پسوندی +*ABC به فرم میانوندی :

AB*C+ ⇒ A+B*C

مثال

حاصل عبارت محاسباتی -,*,-,9,3,8,/,+,1,6,14 کدام است؟

پاسخ:

14 , **6 , 1** , + , / , 8 , 3 , 9 , - , * , -
14 , 7 , / , 8 , 3 , 9 , - , * , -
2 , 8 , **3 , 9** , - , * ,-
2 , **8** , **-6** , * , -
2 , **-48** , -
50

مثال

اگر Postfix عبارتی، AB/C-DE*+AC*- باشد، چند جفت پرانتز در عبارت infix آن قرار دهیم تا postfix آن به صورت، ABC-D+/EA-*C* شود؟

پاسخ: حاصل infix عبارت داده شده برابر A/B-C+D*E-A*C است که با قرار دادن ۲ پرانتز به صورت A/(B-C+D)*(E-A)*C در می آید. postfix آن به صورت ABC-D+/EA-*C* است.

تبدیل از فرم prefix به infix

مثال

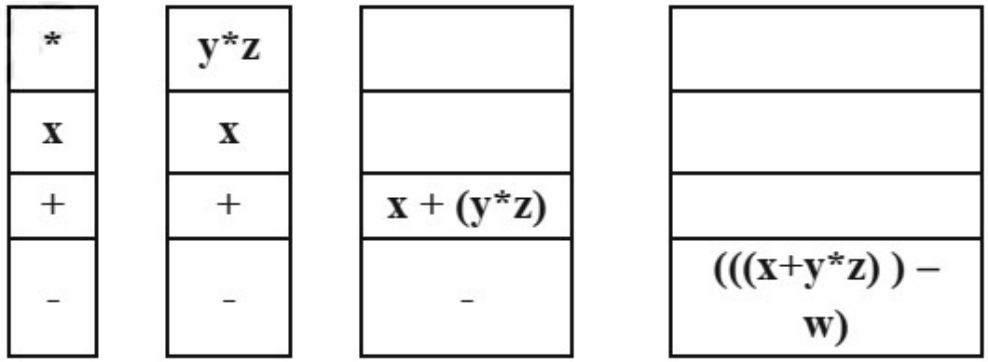
عبارت prefix زیر را به infix تبدیل کنید؟

+ – * ↑ ABCD / E / F + GH

پاسخ:

+ - * **A^B** CD/E/F +GH
+ - * **A^B** CD/E/F **G+H**
+- **A^B*C** D/E/F **G+H**
+-**A^B*C** D/E **F/(G+H)**
+-**A^B*C** D **E/(F/(G+H))**
+ **A^B*C-D** **E/(F/(G+H))**
A^B*C-D + E/(F/(G+H))

مثال عبارت پیشوندی -+x*yzw به فرم میانوندی تبدیل نمایید.(به کمک پشته)



پاسخ:

برای تبدیل به کمک پشته، عملگرهای عبارت را از چپ به راست در پشته قرار داده تا بـه دو عملونـد برسـیم . در ایـن حالـت عملگر بالای پشته را بر روی آنها اعمال کرده و نتیجه را در پشته قرار می‌دهیم. اگر بعد از عملوندی، یک عملگـر بـود آن را در پشته قرار می‌دهیم. اگر در حالتی در بالای پشته دو عبارت قرار داشت که قبل از آنها عملگری بود، ابتدا عملگـر را بـر روی آن دو عبارت اعمال کرده و سپس ورودی‌های بعدی را پردازش می‌کنیم.

تبدیل از فرم Postfix به Prefix

برای این تبدیل ابتدا عبارت postfix را به infix تبدیل کرده و سپس حاصل را به prefix تبدیل می‌کنیم. البته می توان مستقیما نیز این تبدیل را انجام داد.

مثال

عبارت پسوندی ABC ^/D*E+ را به فرم پیشوندی تبدیل نمایید.

پاسخ: ابتدا عبارت را به فرم میانوندی تبدیل می‌کنیم:

ABC ^/D*E+ $\Rightarrow$ A**B^C**/D*E+ $\Rightarrow$ **A/B^C** D*E+ $\Rightarrow$ **A/B^C* D**E+ $\Rightarrow$ **A/B^C* D+E**

حال عبارت میانوندی حاصل را به فرم پیشوندی تبدیل می‌کنیم:

A/B^C*D+E $\Rightarrow$ A/**^BC***D+E $\Rightarrow$ **/A^BC***D+E $\Rightarrow$ ***/A^BCD**+E $\Rightarrow$ **+*/A^BCDE**

مثال

معادل پیشوندی عبارت AB/C*D+ چیست؟ (بدون تبدیل به عبارت میانوندی)

پاسخ: می توان عبارت پسوندی را مستقیما به عبارت پیشوندی تبدیل کرد. یعنی عملگر بعد از عملوندها را به قبل از آنها بـرد.

به طور نمونه عبارت AB/ را به فرم /AB نمایش داد.

AB/C*D+ $\Rightarrow$ **/AB** C*D+ D+ ***/ABC** $\Rightarrow$ $\Rightarrow$ **+*/ABCD**

تبدیل از فرم Prefix به Postfix

ابتدا عبارت prefix را به infix تبدیل کرده و سپس عبارت میانوندی حاصل را به Postfix تبدیل می‌کنیم.

مثال

عبارت پیشوندی /*+ABC-DE به فرم پسوندی تبدیل نمایید.

پاسخ: ابتدا عبارت را به فرم میانوندی تبدیل می‌کنیم:

/*+ABC-DE $\Rightarrow$ /***(A+B)**C-DE $\Rightarrow$ /**((A+B)*C)**–DE

$\Rightarrow$ /**((A+B)*C)(D-E)** $\Rightarrow$ **((A+B)*C)/(D-E)**

حال عبارت میانوندی حاصل را به فرم پسوندی تبدیل می‌کنیم:

((A+B)*C) / (D-E) E) -*C) / (D**AB**+($\Rightarrow$

$\Rightarrow$ **AB+C*** / (D-E) $\Rightarrow$ **AB+C*** / **DE-** /-**AB+C*DE** $\Rightarrow$

مثال

عبارت پیشوندی - +*/ABCD ^/EF-GH را به فرم پسوندی تبدیل نمایید.

پاسخ: می توان مستقیما و بدون استفاده از فرم میانوندی، این تبدیل را انجام داد. کافی است عملگر قبل از عملوندها را به بعد از آنها برد. به طور نمونه عبارت AB/ را به فرم /AB نمایش داد.

- + */ABCD^/EF-GH ⇒ - + ***AB**/CD^/EF-GH ⇒ - + **AB/C***D^/EF-GH ⇒

⇒ - **AB/C*D**+ ^/EF-GH ⇒ - **AB/C*D**+ ^**EF**/-GH ⇒ - **AB/C*D**+ ^**EF**/ **GH**- ⇒

⇒ - **AB/C*D**+ **EF/GH**-^ ⇒ **AB/C*D+EF/GH**-^-

الگوریتم تبدیل عبارت infix به postfix توسط پشته

عبارت داده شده میانوندی را Q و عبارت حاصل در فرم پسوندی را P در نظر می گیریم. ابتدا یک پرانتز باز در پشته push کرده و یک پرانتز بسته به انتهای عبارت Q اضافه می کنیم و عبارت Q را از چپ به راست پیمایش کرده و عملیات زیر را انجام می دهیم:

۱- با برخورد به عملوند، آن را به عبارت P اضافه می کنیم.

۲- با برخورد به پرانتز باز، آن را در پشته push می نماییم.

۳- با برخورد به عملگر، آن را در پشته push می نماییم. البته اگر به عملگری برسیم که اولویت عملگر بالای پشته از این عملگر بیشتر یا مساوی بود، ابتدا عملگر بالای پشته را pop کرده و به P اضافه می کنیم و سپس عملگر مورد نظر را در پشته push می نماییم.

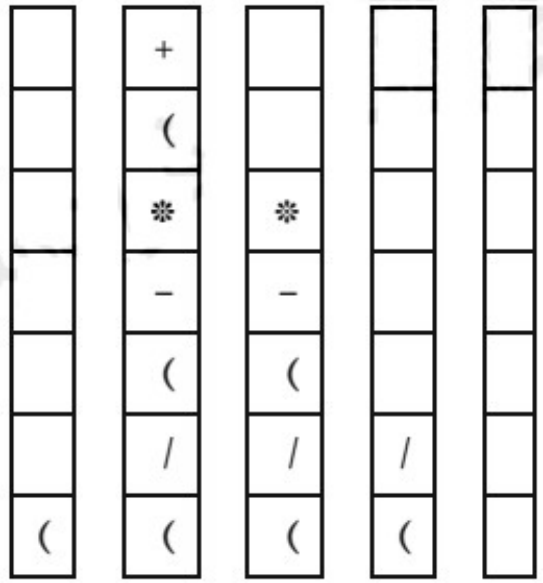
۴- با برخورد به پرانتز بسته، عملگرهای بالای پشته را تا رسیدن به یک پرانتز باز، pop کرده و به P اضافه می نماییم و پرانتز باز داخل پشته را حذف می نماییم.

۵- در نهایت با برخورد به پرانتزی که در ابتدا به انتهای Q اضافه کرده بودیم، عملگر و پرانتز موجود در پشته را pop کرده و عملیات پایان می یابد.(پشته خالی است)

مثال

عبارت میانوندی (A / (B-C* (D+E)) را به کمک پشته به معادل پسوندی تبدیل نمایید.

پاسخ: ابتدا یک پرانتز باز در پشته push کرده و یک پرانتز بسته به آخر عبارت میاوندی اضافه می کنیم و طبق الگوریتم بالا عملیات را انجام می دهیم:



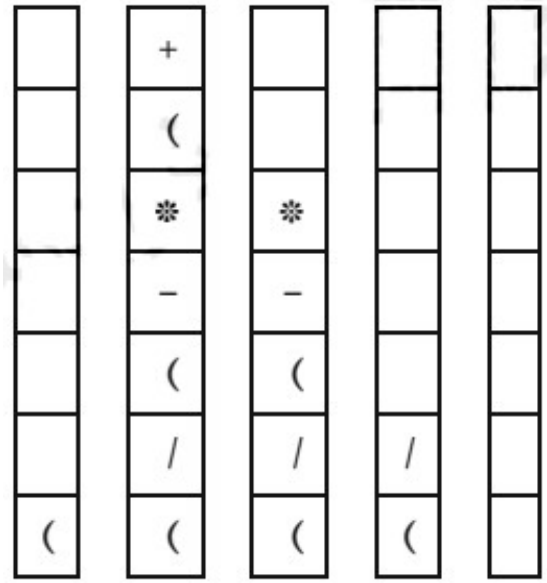
مقدار عبارت P در هر یک از حالت ها در زیر نشان داده شده است:

P = ABCDE

P = ABCDE+

P = ABCDE+*-

P = ABCDE+*-/



الگوریتم محاسبه یک عبارت به فرم Postfix توسط پشته

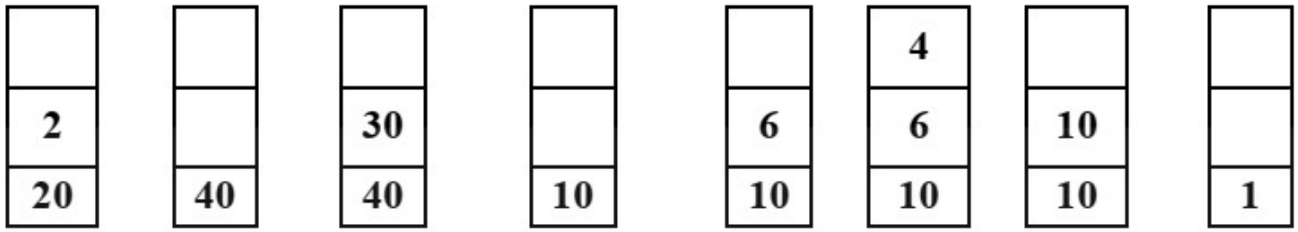
برای محاسبه یک عبارت که به فرم postfix داده شده، ابتدا یک پرانتز بسته نگهبان در انتهای عبارت اضافه می کنیم و به کمک پشته ارزیابی را انجام می دهیم. هنگامی که به پرانتز بسته در عبارت برسیم، نتیجه نهایی در پشته موجود است.

مثال

حاصل عبارت /,+,6,4,-,30,*,2,20 را بدست آورید.(به کمک پشته)

پاسخ:

در ابتدا یک پرانتز بسته به انتهای عبارت داده شده اضافه می کنیم. سپس مقدار 20 و 2 را در پشته push کرده و با رسیدن به عملگر ضرب، آن دو را از پشته pop کرده و نتیجه اعمال ضرب بر روی آنها ، یعنی 40 را در پشته ، push می کنیم و عملیات را به همین نحوه ادامه داده تا به پرانتز بسته در انتهای عبارت برسیم. در این حالت نتیجه در پشته خواهد بود.



در نهایت با 8 عدد push و 8 عدد pop به نتیجه نهایی رسیدیم. (البته push نتیجه نهایی در پشته را در نظر نگرفتیم.)

- تعداد pop ها برای تبدیل یک عبارت postfix به فرم infix ، دو برابر تعداد عملگرها می باشد.

- تعداد push ها برای تبدیل یک عبارت postfix به فرم infix ، دو برابر تعداد عملگرها می باشد.

کاربرد پشته در زیر برنامه های بازگشتی

مثال

خروجی زیر برنامه زیر، به ازای فراخوانی f (1) چیست؟

```
f (int  x){
    if (x==3)  exit( );
   else{
          x=x+1;
          f(x);
          cout<<x;
          cout<< 'a';
    }}
```

پاسخ: ترتیب فراخوانی ها به صورت f(1) => f(2) => f(3) بوده و از آنجا که دستورات بعد از فراخوانی بازگشتی (دستورات چاپ در این مثال)، در پشته قرار می گیرند داریم:

cout<<3
cout<<'a'
cout<<2
cout<<'a'

بنابراین خروجی برابر 3a2a خواهد بود.

مثال

خروجی زیر برنامه زیر، به ازای فراخوانی f (1,4,7) چیست؟

```
f ( int  x , int  y , int  z ){
    if  (x==3)  exit( );
    else{
            x++;  y++;  z++;
            f (x,y,z);
            cout<< x;
          cout<< y;
          cout<< z;
        }
}
```

پاسخ: ترتیب فراخوانی ها به صورت مقابل است: f (1,4,7) => f (2,5,8) => f (3,6,9)

دستورات بعد از فراخوانی بازگشتی (دستورات چاپ در این مثال)، در پشته قرار می گیرند:

cout<< 3
cout<< 6
cout<< 9
cout<< 2
cout<< 5
cout<<8

بنابراین خروجی برابر 369258 خواهد بود.