# اصول ساختمان داده ها در c++

ساختمان داده ها شاخه ای از مهندسی نرم افزار است که به مطالعه انواع ساختارهای داده و نیز الگوریتم های مربوط به آنها می پردازد.

گلزاری پور

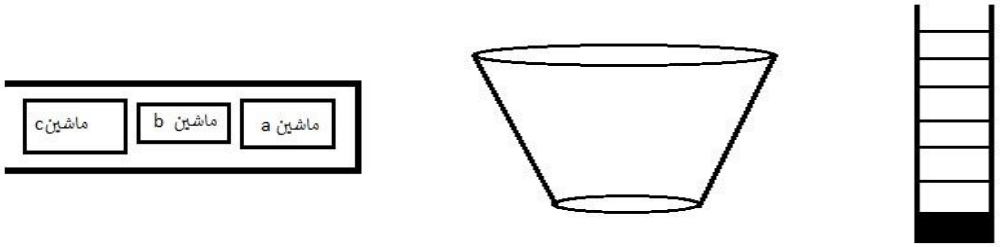
آموزشکده فنی و حرفه ای شهید بهشتی کرج

## پشته[1]

ساختاری است ترکیبی که در آن درج و حذف عناصر تنها از یک سمت آن انجام می شود.

سمتی از پشته که درج و حذف عناصر از آن سمت صورت می گیرد و بالای پشته[2] نامیده می شود . اگر $s=\{a_0 , a_1 , \dots , a_{n-1}\}$ $a_0$ عنصر پائینی و $a_n$ عنصر بالایی پشته است و $a_{i+1}$ در بالای $a_i$ قرار می گیرد.

مثال پارکینگ یکطرفه




## پیاده سازی پشته:

**A.** ساختار داده که شامل موارد زیر است.

۱- مقدار ثابت Maxsize جهت تغیر حداکثر طول پشته.

۲- متغیر Top ، با مقدار اولیه "-1 " جهت مشخص کردن اندیس عنصر بالای پشته

۳- آرایه ای به طول Maxsize که آن را Stack می نامیم . به عنوان مثال آرایه از نوع صحیح :

```
int Maxsize = 10;
int top = -1 ;
int Stack[maxzise];
```

[1] Stack
[2] Top of stack

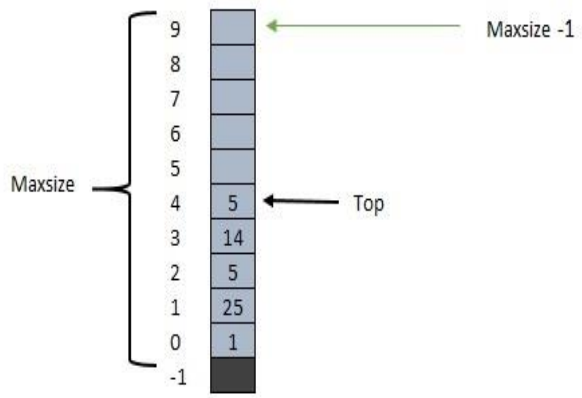
B. توابع چهار گانه جحت انجام اعمال درج و حذف در پشته:

تابع IsFull جهت بررسی پر بودن پشته

۱- شرط پر بودن پشته این است که متغر Top برایر باشد با Maxsize-1

```
int IsFull(){
if(Top==Maxsize-1)
return 1 ;
        else
        return 0;

}
```




۲- تابع IsEmpty جهت بررسی خالی بودن پشته:

شرط خالی بودن پشته این است که Top برابر 1 - باشد.

```
int IsEmpty(){
if(Top==-1)
return 1;
else
return 0;
}
```

۳- تابع Push برای درج عنصر x در پشته است.

ابتدا یک واحد به متفیر Top اضافه می شود سپس مقدار x را در موقیعت جدید اضافه میکند

```
void Push(int x)
{
if (Isfull())
cout<<"stack is full. " ;
else
Stack [++Top]=x;
}
```

۴- تابع Pop جهت انجام عمل حذف از پشته :

ابتدا باید عنصر های پشته یعنی Stack[Top] را در متغیر مرجع x قرار داد.

سپس باید Top را یکی کاهش داد.

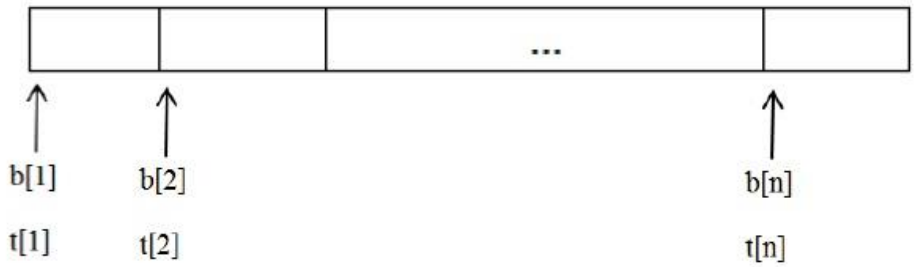
```
void Pop(int &s)
{
        if(IsEmpty())
cout << “Stack is Empty “ ;
else
  x = Stack[Top--];
return x ;
}
```

متغیر ورودی &x، متغیر نوع مرجع نام دارد و مانند ارسال ارجائی پارامتر ها به تابع است x یک متغیر

صحیح بوده و &X آدرس متغیر x در حافظه است.

# پشته چندگانه

ولی اگر بیش از دو پشته در یک آرایه پیاده سازی شود، در این حالت روش قبلی قابل اجرا نیست. در این حالت برای ساخت n پشته در آرایه S[1…m]، آرایه را n قسمت می کنیم. برای هر پشته i، b[i] به پایین ترین عنصر و t[i] به بالاترین عنصر اشاره می کند و اگر b[i]=t[i] باشد آنگاه پشته iام خالی است.



## کاربرد های پشته

- ✓ فراخوانی توابع در برنامه
- ✓ ارزیابی عبارت محاسباتی و منطقی
- ✓ تبدیل مبنای عدد
- ✓ تشخیص تقارن در پشته ها

روش های مختلف نمایش عبارات محا سباتی و ریاضی :

۱. روش میانوندی یا infix : در این روش هر عملگر بین عملوندهای خود قرار می گیرد. مثل a+b
۲. روش پسوندی یا postfix : در این روش عر عملگر بعد از عملوند های خود قرار می گیرد مثلاً نمایش پسوندی برای عبارت a*b به صورت *ab میباشد.
۳. روش پیشوندی یا prefix : در این روش عملگر قبل از عملوند های خود قرار می گیرد به عنوان مثال *ab نمایش پیشوندی برای عبارت a*b است

## مثال

عبارت میانوندی a*b/c را به صورت پسوندی و پیشوندی نمایش دهید.

postfix : ab*c/
prefix : /*abc

نشانه های عبارت میانوندی را از چپ به راست مورد بررسی قرار می دهیم:

نشانه ها عبارت اند از : عملگرها و عملوندها ، پرانتز باز و بسته و یا نشانه پایان رشته.

a|b-c+(d*e)/f '/0'

با توجه به نوع نشانه دیده شده حالات زیر وجود دارد:

۱. اگر نشانه عملوند است مستقیم به خروجی فرستاده شود.
۲. اگر نشانه پرانتز بسته است عناصر پشته را تا رسیدن به اولین پرانتز به خروجی می فرستیم.
۳. اگر نشانه پرانتز باز است بلا فاصله آن را در بالای پشته درج کنیم
۴. اگر نشانه عمگر است عملگر جدید را با عملگر بالای پشته( در صورت وجود ) مقایسه میکنیم که دو حالت زیر پیش خواهد آمد:

الف ) اگر پشته خالی باشد یا اولویت عملکر جدید بیشتر باشد عملگر در پشته درج شود.

ب) اگر عملگر جدید دارای اولویت برابر یا پایین باشد عناصر پشته را تا رسیدن به پرانتز باز و یا عملگری که اولویت آن از اولویت آن از اولویت عملگر جدید پائین تر است خارج کنیم و به خروجی ارسال کنیم سپس عملگر جدید را در پشته درج نمائیم.

**مثال**

تبدیل عبارت میانوندی

a*(b+c)/d

| نشانه | پشته | خروجی |
|---|---|---|
| a | | a |
| * | * | a |
| ( | *( | a |
| b | *( | a b |
| + | *(+ | a b |
| c | *(+ | a b c |

| | | |
|---|---|---|
| ) | *( | abc+ |
| / | / | abc=* |
| d | / | abc+*d |

الگوریتم ارزیابی یک عبارت پسوندی : منظوراز ارزیابی یک عبارت محاسبه حامل آن به ازای مقادیر عملوندهای آن است.

۱. نشانه های عبارت را ازچپ به راست مورد بررسی قرار دهیم.

۲. با توجه به نوع نشانه سه حالت وجود دارد:

.الف) اگر نشانه عملوند است آن را در بالای پشته درج کنیم

ب)اگر نشانه یک عملگر است دراین صورت باتوجه به اینکه عملگر دو عملوندی یا تک عملوندی باشد.به ترکیب دو یا یک نشانه از پشته خارج شده به عملگر مورد نظر بر روی آنها اعمال شود نتیجه در پشته درج می شود.

ج) نشانه کاراکتر پایان رشته است . معنی آن این است که ارزیابی عبارت به پایان رسیده است.

دراین حالت پشته باید تنها دارای یک مقدار باشد که جواب مسئله است . با pop نمودن این مقدار باید پشته خالی شود.

نحوه ارزیابی عبارت پسوندی : -*ac+*de-c/ab باتوجه به الگوریتم بیان شده محاسبه کنید:

راهنمایی : فرض کنید هر بار که مقداری را محاسبه می کنیم آن را در متغیر T ذخیره می کنیم.

**نکته**

در موقع پوش کردن باید عنصر مشخص باشد x مقدار اصلی آن آدرس آن چیست ولی در پاپ کردن لازم نیست آخرین عنصر خودش ظاهر می شود.

| نشانه | نوع | عملگر | پشته |
|---|---|---|---|
| a | عملوند | push (a) | a |
| b | عملوند | push (b) | ab |
| / | عملگر | Pop $=x_2=b$ | |
| | | Pop$=x_1=a$ | |
| | | $T_1=a/b$ | |
| | | Push$=(T_1)$ | $T_1$ |

# اولویت عملگرها

۱- ()

۲- not و توان و – (قرینه)

۳- and و mod و div و / و *

۴- or و - و +

۵- < و =< و <> و = و => و >

مثال: ترتیب عملیات در عبارت زیر را مشخص نمائید.

$a+b*c^a-b/c$

جواب:

$((a+(b*(c^a)))-(b/c))$

## تبدیل عبارت میانوندی به پسوندی و پیشوندی

۱- ابتدا عبارت را طبق اولویت پرانتزگذاری کنید.

۲- سپس برای تبدیل به پسوندی هر عملگر را به سمت راست پرانتز بسته خودش منتقل کنید.

۳- برای تبدیل به پیشوندی هر عملگر را به سمت چپ پرانتز باز خودش منتقل کنید.

۴- سپس پرانتزها را حذف کنید.

مثال: فرم پسوندی و پیشوندی عبارت میانوندی زیر را بیابید.

$$a + b * c \hat{} a - b/c$$

جواب:

$$\left(\left(a + (b * (c \hat{} a))\right) - \left(\frac{b}{c}\right)\right)$$

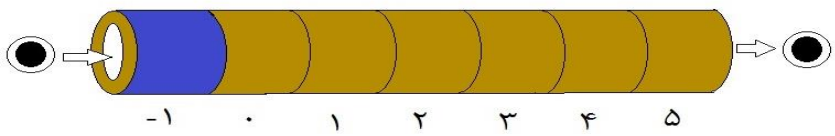
فرم پسوندی:

$$abca \hat{} * + bc/ -$$

فرم پیشوندی:

$$- + a * b \hat{} ca/bc$$

# صف[3]




ساختاری است ترکیبی که در آن درج عناصراز انتها وحذف آنها از ابتدا انجام می شود. ابتدا وانتهای صف به ترتیب با دو متغیر front و rear نشان داده می شود .اگر $Q = \{a_0,a_1,...,a_{n-1}\}$ صفی ازعنصر باشد آنگاه $s_0$ را عنصرابتدایی و $a_{n-1}$ را عنصر انتهایی صف می نامند .

از تعریف چنین استنباط می شود که در صف اولین عنصری که وارد می شود. اولین عنصری است که خارج می شود به همین علت ، صف را می توان به عنوان یک لیست FIFO تلقی کرد.

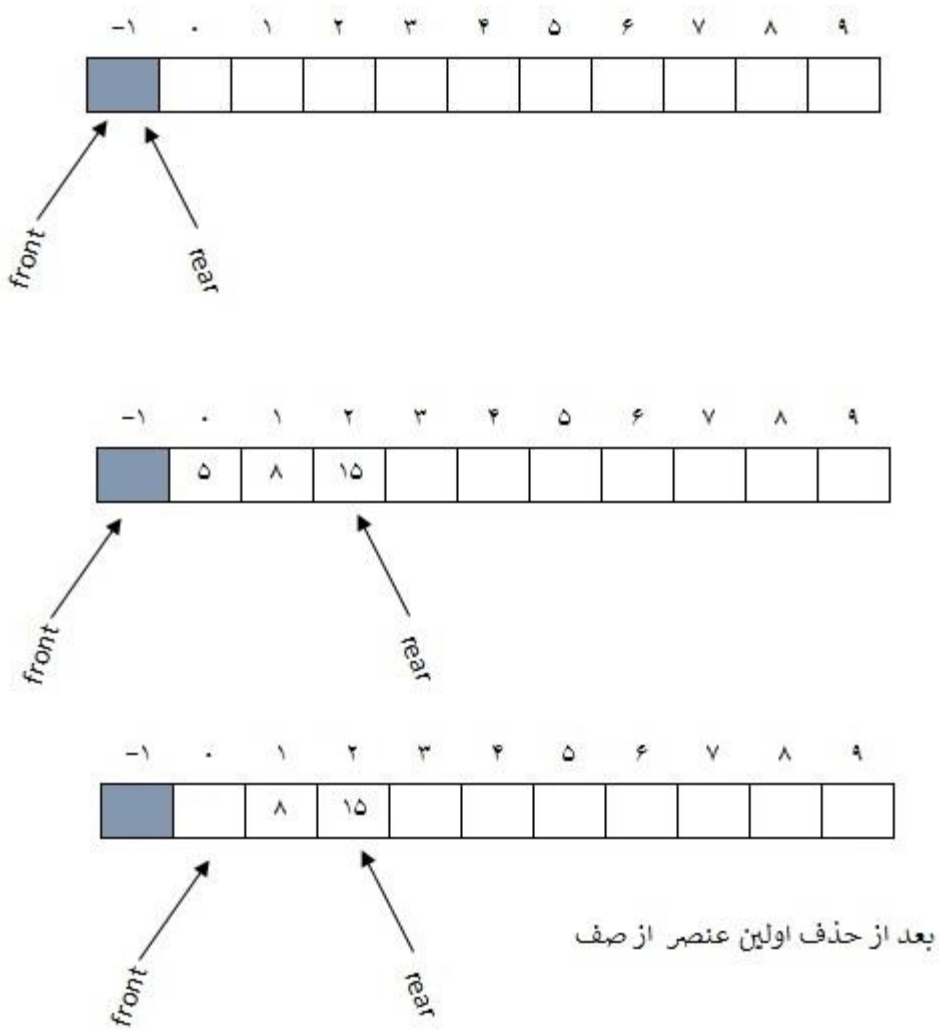
بر خلاف پشته صف ساختاری دو طرفه است ، ابتدا و انتهای صف با دو متغیر front و rear نشان داده می شود . موقیعت front بر یک موقیعت عقب تر از یک موقیعت اولین عنصر و rear دقیقاً به موقیعت آخرین عنصر صف اشاره می کند.

شرط خالی بودن صف (n=0) عبارت است از front = rear = -1 باشد که شرط اولیه می باشد.

**نکته**

با هر عمل درج عنصر در صف یک واحد به متغیر rear اضافه می شود و با هر عمل حذف عنصر یک واحد به متغیر front اضافه می شود.

[3] Queue




## مقایسه پشته و صف

۱. هر دو ساختار های ترکیبی داشته و حالت ویژه ای از آرایه هستند.

۲. پشته از حالت LIFO[4] ولی صف از حالت FIFO[5] می باشد.

۳. پشته لیستی یک طرفه ولی صف لیستی دو طرفه می باشد.

۴.

[4] Last In First Out ( LIFO)
[5] First In First Out (FIFO)

## پیاده سازی صف

A- تعریف ساختار داده که شامل موارد زیر است:

- مقدار ثابت Maxsize جهت تعیین حداکثر طول صف.
- دو متغیر front , rear با مقدار اولیه (۱-) جهت مشخص نمودن اندیس ابتدائی و انتهائی صف
- آرایه به طول Maxsize به نام Queue و از نوع صحیح

```
Const  int Maxsize = 10;
int front = rear = -1 ;
Queue[Maxsize];
```

B- توابع چهار گانه زیر جهت انجام اعمال درج و حذف عصر در صف .

۱- تابع IsFull جهت بررسی پر بودن صف

```
int IsFull(){
if ( rear == Maxsize -1 )
return 1 ;
else return 0;
```

۲- تابع IsEmpty جهت بررسی خالی بودن صف.

```
int IsEmpty( ){
if ( front == rear )
return 1 ;
else return 0;
}
```

۳- تابع add جهت درج عنصر x در انتهای صف

```
void Add (int x ){
if ( IsFull( ) )
cout << " Queue is Full." ;
else Queue[ ++rear ] = x;
}
```

۴- تابع Del جهت انجام عمل حذف از ابتدای صف.

```
void Del (int &x)
{
if ( IsEmpty() )
cout<<" Queue is Empty";
else
x = Queue[++front];
}
```

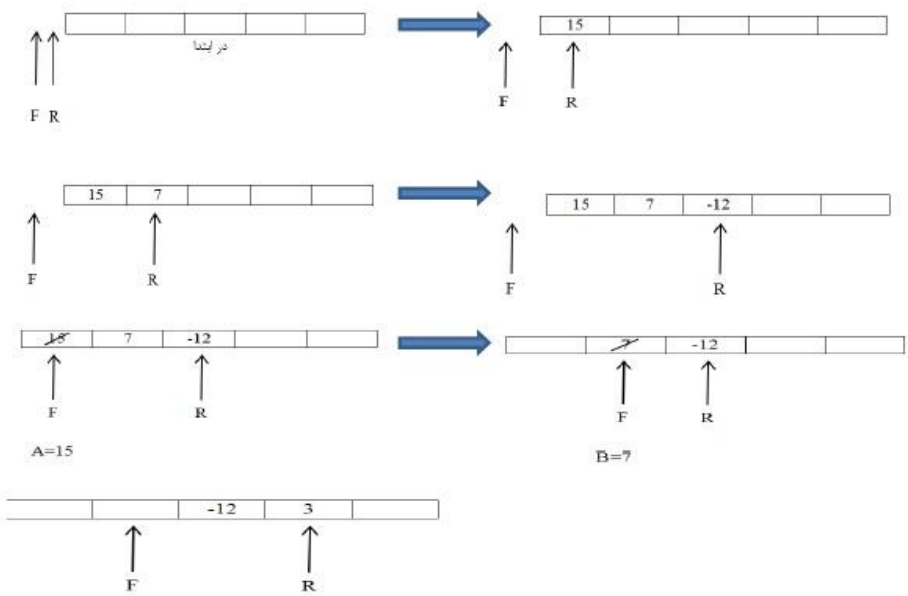
سوال : تابع Del مقدار x را چگونه به برنامه اصلی برمی گرداند ؟

اول یک واحد اضافه بعد عنصر واقع در آن به x منتقل می شود . زیرا متغیر Front به یک موقیعت عقب تر از اولین عنصر اشاره می کند.

پیچیدگی زمانی الگوریتم های درج و حذف در صف دارای زمان ثابت O(1) است چون برای این اعمال از دستورات شرطی و جایگزینی استفاده می شود که زمان ثابتی طول میکشد.

مثال: عملیات زیر را به ترتیب از چپ به راست روی شکل نشان دهید.

add (15),add (7),add (-12),del (A),del (B),add (3)



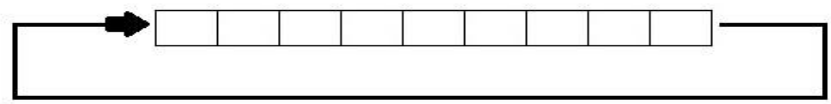
## صف حلقوی[6]

صف حلقوی صفی است که در آن موقعیت های ابتدائی و انتهایی بر هم منطبق هستند.

**نکته**

در سطح حلقوی اگر rear = Maxsize -1 باشد ولی در امتداد صف تعدادی خانه خالی وجود داشته باشد انگاه عنصر جدید در این خانه ها می تواند درج شود.

یکی از معایب صف خطی این است که با انجام عملیات حذف نقطه ابتدائی صف به تدریج به سمت راست تغییر مکان می یابد . به طوری که به تدریج مکانهای ابتدای صف بلا استفاده باقی مانده و نمی توان عنصر جدیدی در آن درج کرد . چون طبق ماهیت صف عناصر جدید نمی توانند در ابتدای صف درج گردند.



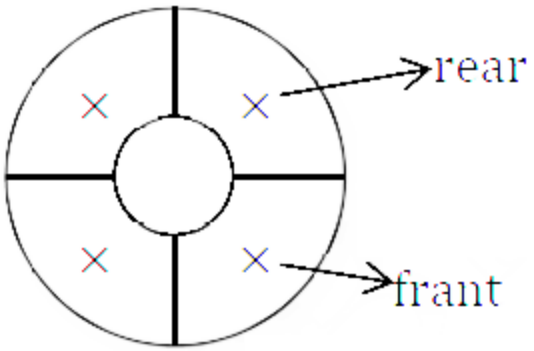
**نکات زیر در سطح حلقوی قابل توجه است.**

۱- مانند سطح خطی front همیشه به یک موقعیت عقب تر از عنصر اشاره می کند.

۲- مانند صف خطی شرط خالی بودن صف این است که front برابر rear باشد

۳- در یک صف حلقوی به طول maxsize که با حرف n هم نشان می دهند. حداکثر تعداد عناصر مجاز برابر maxsize- 1 است زیرا در غیر این صورت در حالتی که صف پر است تعداد عناصر برابر maxsize و در نتیجه رابطه front=rear است . به عبارت دیگر نمی توان تمایز بین حالات پر بودن و خالی بودن صف قائل شد که این مسئله باعث بروز مشکل در پیاده سازی توابع add و delete می گردد .

[6] Circular Queue




۴- در صف حلقوی مقادیر اولیه برای متغیرهایfront و rear صفر است.

۵- در صف حلقوی بر خلاف صف خطی شرط پر بودن صف این است که front=(rear+1)%maxsize باشد.

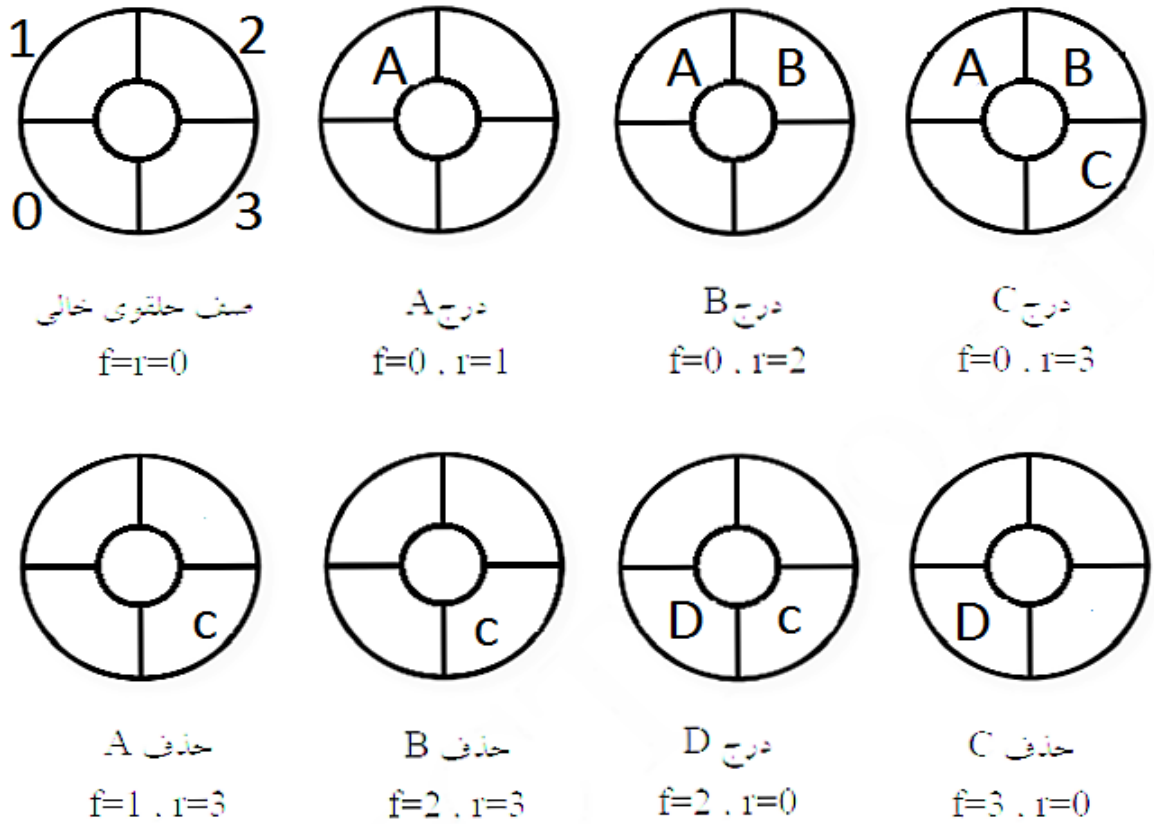
۶- در صورتی که صف پر نباشد برای عمل درج باید به متغیر rear یک واحد اضافه کرده و سپس باقیمانده تقسیم آن را بر maxsize محاسبه نمائیم حاصل محاسبه موقعیت عنصر جدید را جهت عمل درج نشان می دهد .

۷- در صورتی که صف خالی نباشد برای عمل حذف باید به متغیر front یک واحد اضافه کرد و پس از آن باقیمانده تقسیم آن را بر maxsize محاسبه نمود . حاصل محاسبه ، موقعیت عنصری که باید حذف شود نشان می دهد .

۸- هر دو عمل درج و حذف در جهت عقربه های ساعت انجام می شود .

**مثال**

عملیات درج و حذف در یک صف حلقوی به طول حداکثر سه عنصر را نمایش می دهد دقت کنید که همیشه باید یک خانه از صف خالی باقی بماند .




حلقوی سطح در حذف و درج توابع

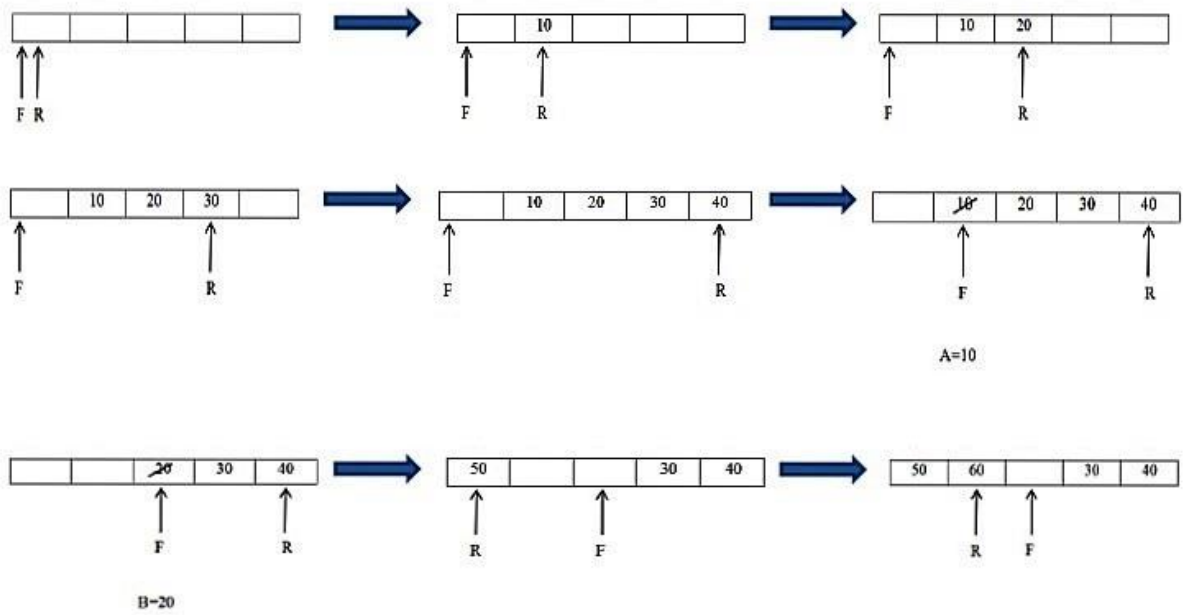
```
Void add (const int x)
{
Int k =(rear+1)% maxsize;
If (front==k)
Cout<<"Queue is full";
Else {rear =k ; Queue[rear]=x;}}
```

تابع Delete

```
Void delete (int & x)
{
If (front==rear)
Cout<<"Queue is empty";
Else{front++ ; front%=maxsize ;  X=Queue[front] ; }}
```

**مثال:** عملیات زیر را به ترتیب از چپ به راست روی شکل نشان دهید.

add(10) ,add(20), ,add(30) ,add(40) ,del(A), ,del(B) ,add(50) ,add(60)




## کاربردهای صف

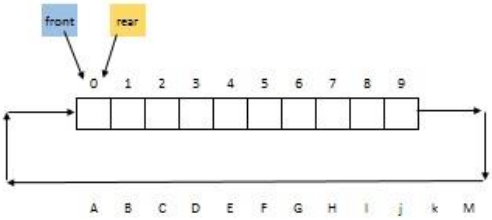
۱- در سیستم عامل جهت پردازش فرآیندها

۲- ذخیره سازی کلیدهای فشرده شده در حین اجرای برنامه ها

۳- در مسائل شبیه سازی جهان واقع

۴- در شبکه های کامپیوتری برای استفاده بهینه از منابع شبکه مثل پرینتر

## مثال صف حلقوی




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | | | | | | | | |
| | A | B | | | | | | | |
| | A | B | C | | | | | | |
| | A | B | C | D | | | | | |
| | A | B | C | D | E | | | | |
| | A | B | C | D | E | F | | | |
| | A | B | C | D | E | F | G | | |
| | A | B | C | D | E | F | G | H | |
| | A | B | C | D | E | F | G | H | I |
| | A | B | C | D | E | F | G | H | I |
| | | | | | | | | | |
| | | B | C | D | E | F | G | H | I |
| | | | | | | | | | |
| | | | C | D | E | F | G | H | I |
| | | | | | | | | | |
| | | | | D | E | F | G | H | I |
| J | | | | D | E | F | G | H | I |
| J | K | | | D | E | F | G | H | I |
| J | K | M | | D | E | F | G | H | I |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| 1 | | add(A) | k=(rear+1)%maxsize | k=(0+1)%10=1 | | F | | | | | 1 | 1 | 0 |
| 2 | | add(B) | k=(rear+1)%maxsize | k=(1+1)%10=2 | | F | | | | | 2 | 2 | 0 |
| 3 | | add(C) | k=(rear+1)%maxsize | k=(2+1)%10=3 | | F | | | | | 3 | 3 | 0 |
| 4 | | add(D) | k=(rear+1)%maxsize | k=(3+1)%10=4 | | F | | | | | 4 | 4 | 0 |
| 5 | | add(E) | k=(rear+1)%maxsize | k=(4+1)%10=5 | | F | | | | | 5 | 5 | 0 |
| 6 | | add(F) | k=(rear+1)%maxsize | k=(5+1)%10=6 | | F | | | | | 6 | 6 | 0 |
| 7 | | add(G) | k=(rear+1)%maxsize | k=(6+1)%10=7 | | F | | | | | 7 | 7 | 0 |
| 8 | | add(H) | k=(rear+1)%maxsize | k=(7+1)%10=8 | | F | | | | | 8 | 8 | 0 |
| 9 | | add(I) | k=(rear+1)%maxsize | k=(8+1)%10=9 | | F | | | | | 9 | 9 | 0 |
| 10 | | add(J) | k=(rear+1)%maxsize | k=(9+1)%10=0 | | | | | | غیرقابل اجرا | 0 | 9 | 0 |
| 11 | | delet(x) | front++;<br>front%=maxsize | | | | | | | | 0 | 9 | 1 |
| | | | | | | | | | | | 0 | | 1 |
| 12 | | delet(x) | front++;<br>front%=maxsize | | | | | | | | 0 | 9 | 2 |
| | | | | | | | | | | | 0 | | 2 |
| 13 | | delet(x) | front++;<br>front%=maxsize | | | | | | | | 0 | 9 | 3 |
| | | | | | | | | | | | 0 | | 3 |
| 14 | | add(J) | k=(rear+1)%maxsize | | k=(9+1)%10=0 | F | | | | | 0 | 0 | 3 |
| 15 | | add(k) | k=(rear+1)%maxsize | | k=(0+1)%10=1 | F | | | | | 1 | 1 | 3 |
| 16 | | add(M) | k=(rear+1)%maxsize | | k=(1+1)%10=2 | F | | | | | 2 | 2 | 3 |

| k | rear | front | maxsize |
|---|---|---|---|
| | 0 | 0 | 10 |