فصل سوم : صف وپشته

اهداف

- آشنایی با پشته
- آشنایی با صف
- ارزشیابی عبارات

پشته

پشته و صف ، حالات خاصی از نوع داده عمومی یعنی لیست های مرتب شده ، می باشند.

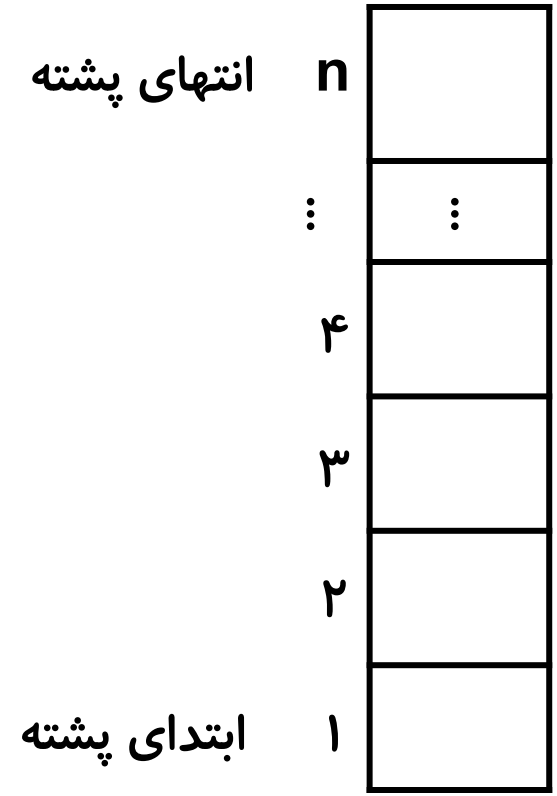
پشته

پشته یک لیست مرتب شده ای است که جایگذاری و حذف از یک سمت آن که top (بالا) نامیده می شود ، صورت می گیرد.

✓در پشته ای مانند $S = a_0,\ldots,a_{n-1}$ ، a_0 عنصر پایینی و a_{n-1} عنصر بالایی می‌باشد.

پشته

خانه های آرایه از ۱ تا n شماره گذاری شده و در کنار آرایه متغیری به نام TOP به عنصر بالایی پشته اشاره دارد



نوع داده `stack[n]`

دامنه تغییرات `TOP: 0 .. n`

مقدار اولیه `TOP=0`

شرط خالی بودن `TOP=0`

شرط پر بودن `TOP=n`

پشته

❖محدودیت کار با پشته ما را ملزم می کند که اگر عناصر *E،D،C،B،A* را به ترتیب به پشته اضافه کنیم ، *E* اولین عنصری خواهد بود که که از پشته حذف می گردد.

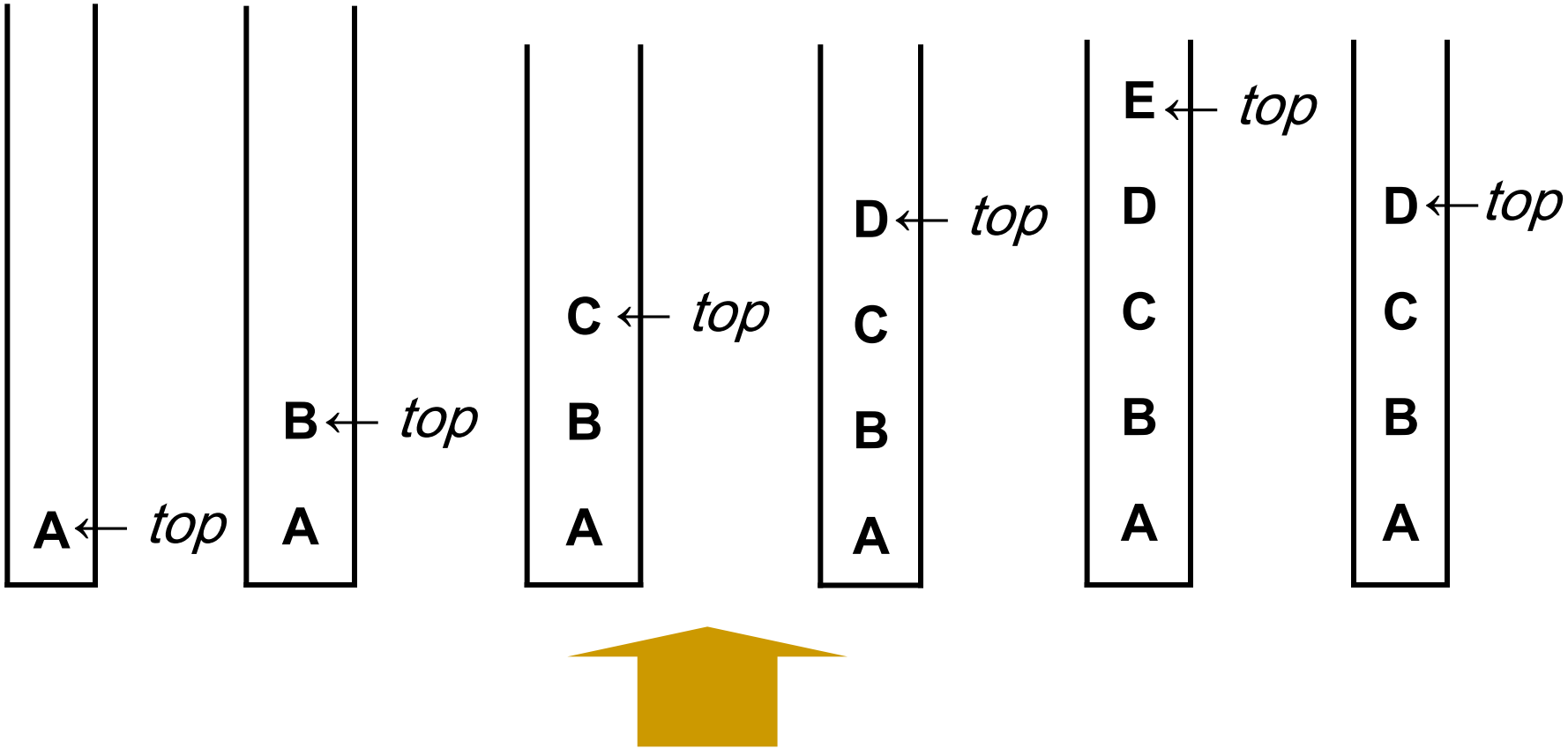
❖از آنجا که آخرین عنصر وارده به پشته ، اولین عنصر حذف شده از آن می‌باشد ، پشته را به عنوان یک لیست *LIFO* (آخرین ورودی ، اولین خروجی) می شناسیم.

مهمترین توابع پشته PUSH و POP

```
void push(int x);
{
        if (top==n)
            cout << “Stack is full”;
        else
        {
            top++;
            stack[top]=x;
        }
}
```

```
int pop();
{
        if (top==0)
            cout << “Stack is empty”;
        else
        {
            int x=stack[top];
            top--;
        }
        return x;
}
```

پشته



حذف و جایگذاری عناصر در یک پشته

استفاده از کتابخانه پشته

برنامه ۱-۳

```
#include <iostream>
#include <stack>

using namespace std;

void showstack(stack <int> gq)
{
    stack <int> g = gq;
    while (!g.empty())
    {
        cout << '\t' << g.top();
        g.pop();
    }
    cout << '\n';
}
```

استفاده از کتابخانه پشته

برنامه ۱-۳

```
int main ()
{
    stack <int> myStack;
    myStack.push(10);
    myStack.push(30);
    myStack.push(20);
    myStack.push(5);
    myStack.push(1);

    cout << "The stack myStack is : ";
    showstack(myStack);

    cout << "\n myStack.size() : " << myStack.size();
    cout << "\n myStack.top() : " << myStack.top();


    cout << "\n myStack.pop() : ";
    myStack.pop();
    showstack(myStack);

    return 0;
}
```

صف (Queue)

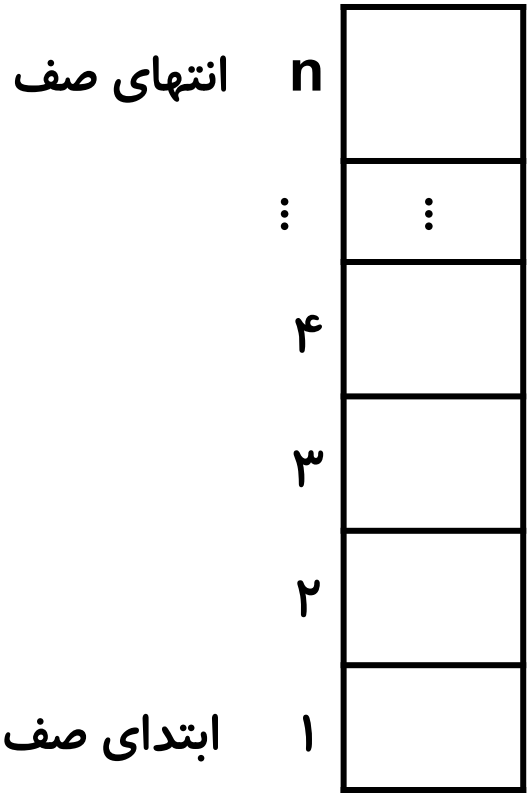
صف

صف یک لیست مرتب است که تمامی جایگذاری آن از یک سمت و تمام حذف های ان از سمت دیگر انجام می شود.

درصف $Q = a_0, a_1, ..., a_{n-1}$ ، a_0 عنصر ابتدا (front) و a_{n-1} عنصر انتها (rear) می‌باشد و a_{i+1} در کنار a_i قرار دارد $(0 \leq i < n-1)$

صف خطی

خانه های آرایه از ۱ تا n شماره گذاری شده و در کنار آرایه به دو اشاره گر نیاز داریم؛ FRONT به عنصر قبل از عنصر ابتدایی اشاره می کند و REAR به آخرین عنصر اشاره می کند



نوع داده queue[n]

دامنه تغییرات FRONT,REAR:0.. n

مقدار اولیه FRONT=REAR=0

شرط خالی بودن FRONT=REAR

شرط پر بودن REAR=n

صف

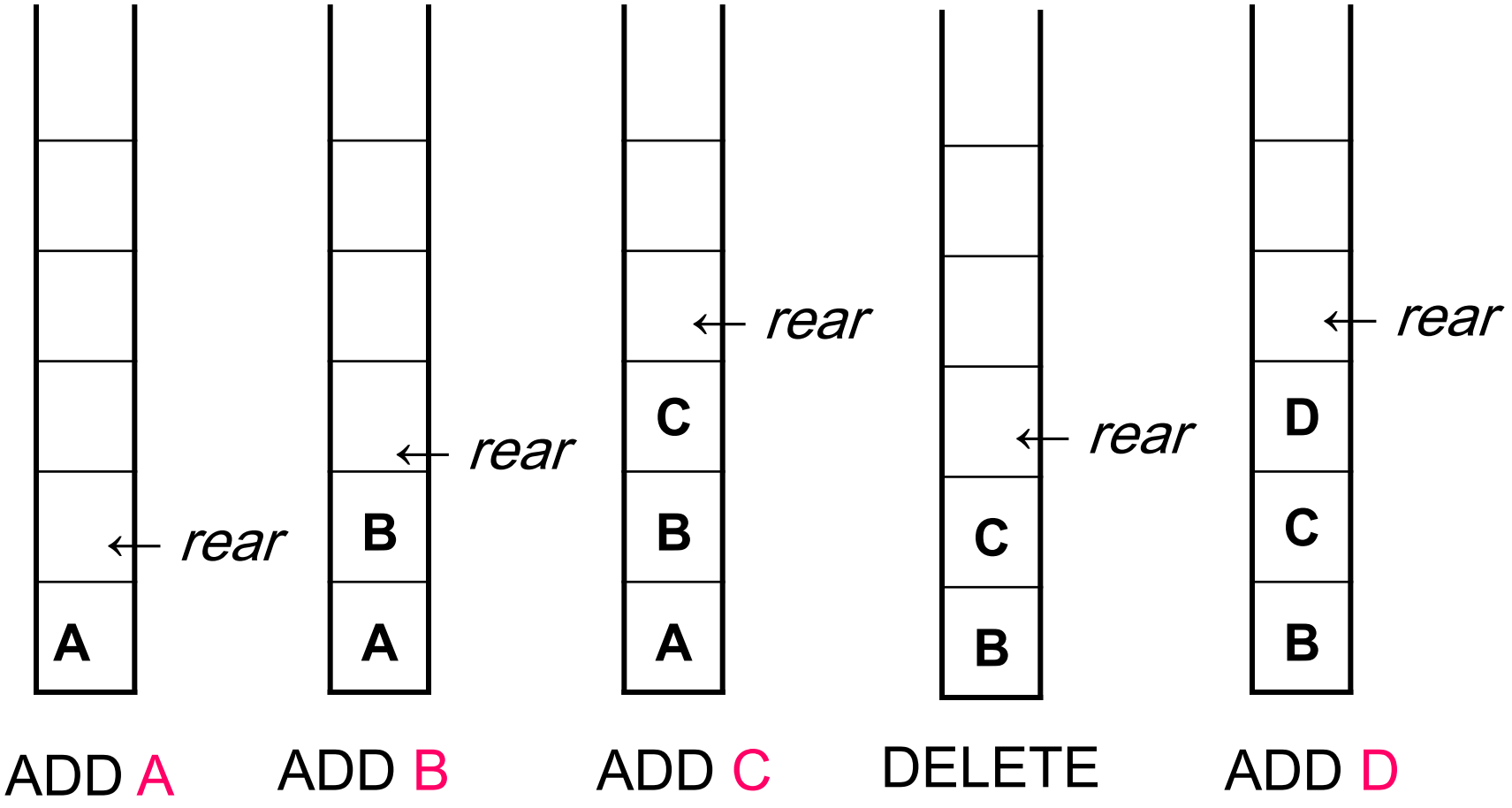
از آنجا که اولین وارد شده به یک صف ، اولین عنصری است که خارج می شود ، صف را به عنوان لیست های FIFO (اولین ورودی، اولین خروجی) در نظر می گیرند.

مهمترین توابع صفADD و DEL

```
void add(int x);
{
        if (rear==n)
            cout << “Queue is full”;
        else
        {
            rear++;
            queue[rear]=x;
        }
}
```

```
int del();
{
        if (front== rear)
            cout << “Queue is empty”;
        else
        {
            front++;
            int x=queue[front];
        }
        return x;
}
```

محدودیت صف این است که ما D،C،B،A را به ترتیب اضافه می کنیم در حالی که A اولین عنصری است که حذف می شود.



درج و حذف عناصر از یک صف

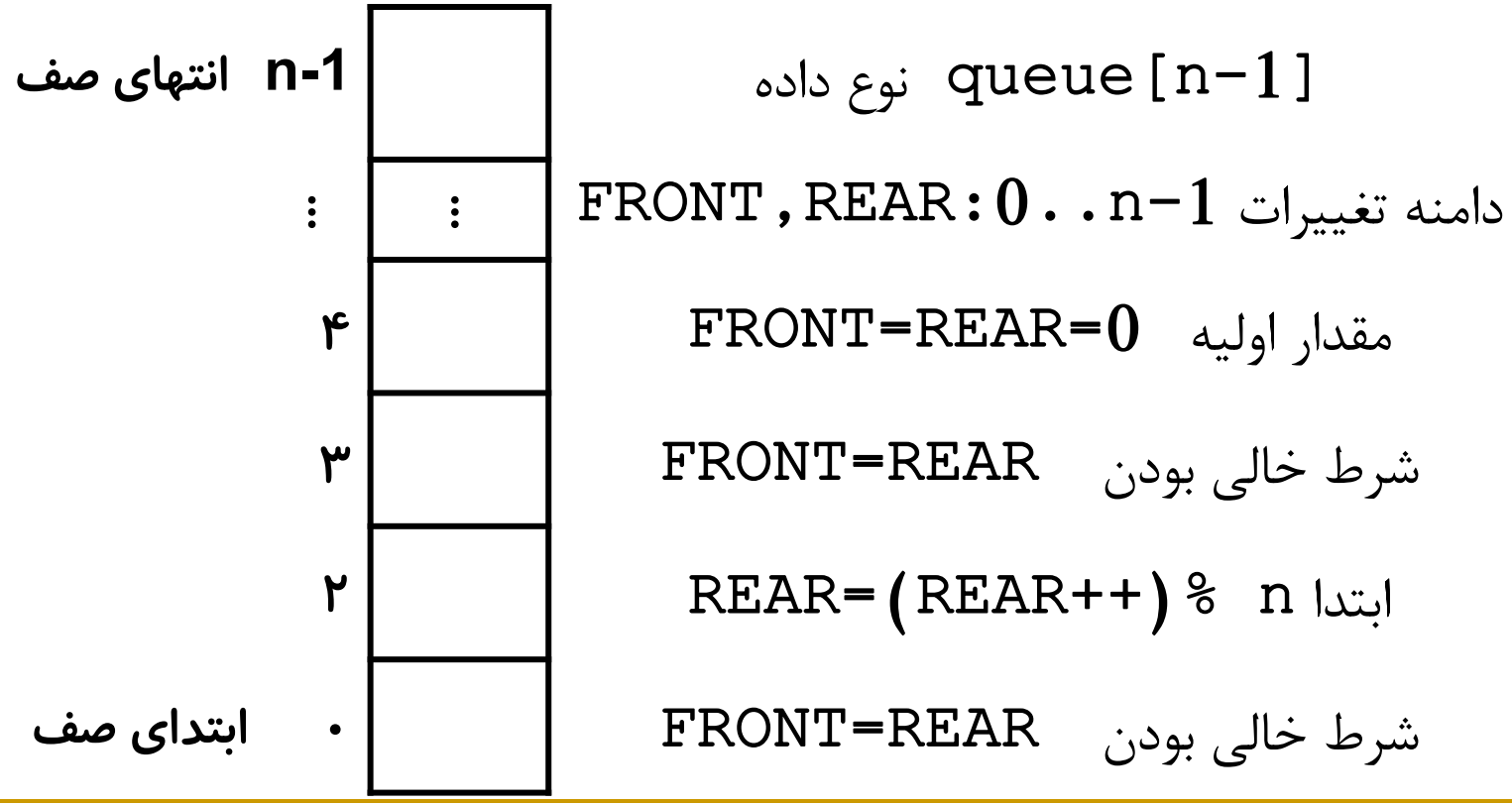
استفاده از کتابخانه صف

برنامه ۲-۳

```
#include <iostream>
 #include <queue>
 using namespace std;
 int main() {
   queue<int> q;
   q.push(100);
   q.push(200);
   q.push(300);
   q.push(400);
   cout << "Size of the queue: " << q.size() << endl;
   cout << q.front() << endl;
   q.pop();
   cout << q.front() << endl;
   q.pop();
   cout << q.front() << endl;
   q.pop();
   cout << q.front() << endl;
   q.pop();
   if ( q.empty() ) {
        cout << "Queue is empty" << endl;
        }
}
```

صف چرخشی

مشکل اصلی صف خطی آن است که فقط یکبار قابل استفاده است و هنگامی که REAR به انتها می رسد نمی توان در صف چیزی را ذخیره کرد.



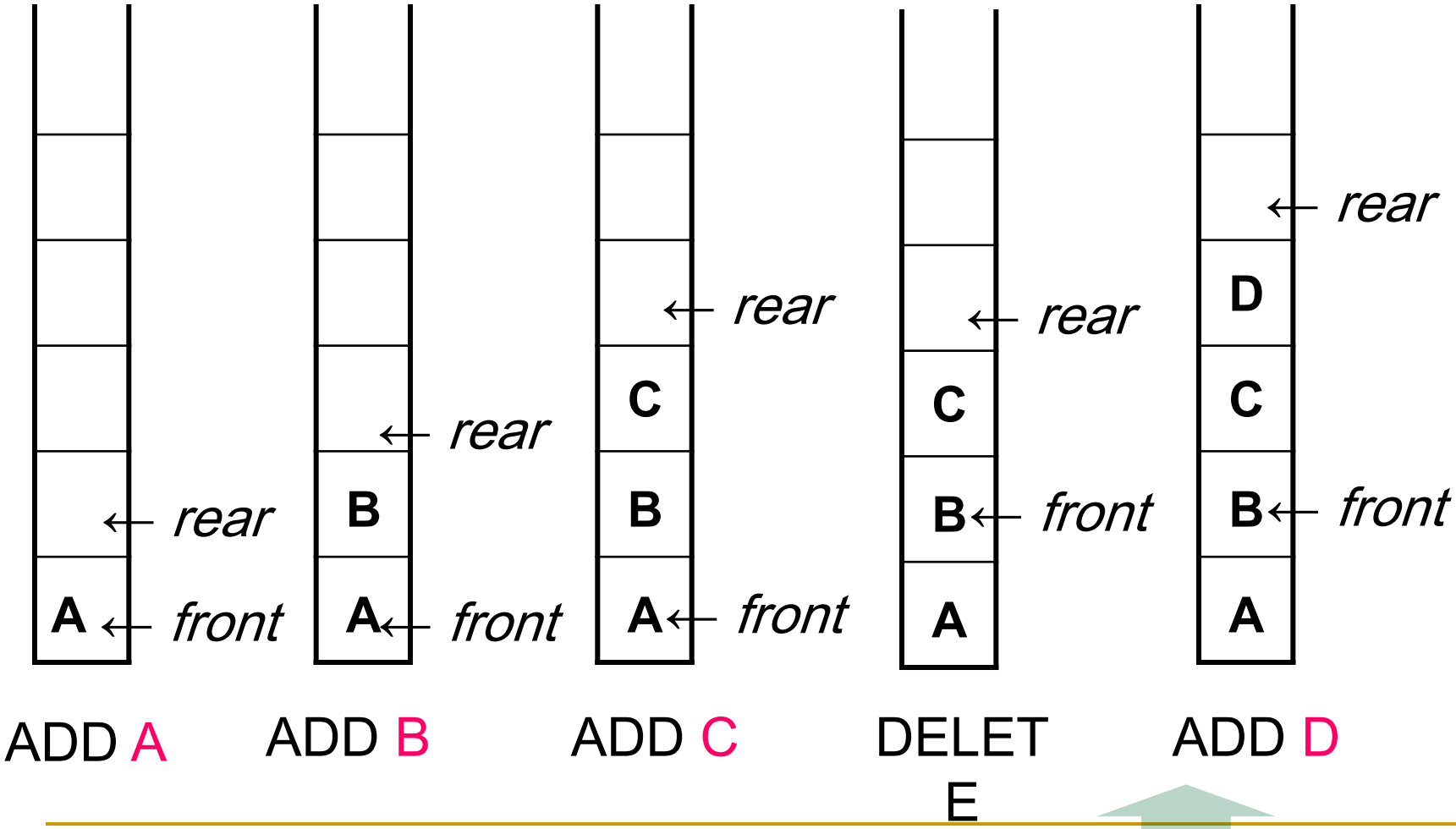
توابع صف چرخشی ADD و DEL

```
void add(int x);
{
        rear=(rear++)% n;
        if (rear==front)
            cout << “Queue is full”;
        else
        {
            queue[rear]=x;
        }
}
```

```
int del();
{
        if (front== rear)
            cout << “Queue is empty”;
        else
        {
            front =(front ++)% n;
            int x=queue[front];
        }
        return x;
}
```

صف چرخشی

در این روش پس از حذف یک عنصر، عناصر جابجا نمیشوند بلکه تنها اندیس نشان دهنده آغاز صف به جلو حرکت میکند.



درج و حذف عناصر از یک صف چرخشی

کاربرد Stack: ارزشیابی عبارات

در هر زبان برنامه سازی برای ارزیابی صحیح عبارات ، به هر عملگر اولویتی نسبت می دهیم. حال در هر جفت پرانتز ، اول عملگرهایی که دارای اولویت بالاتر هستند ، ارزشیابی می شوند .

اولویت عملگرها

- شکل مقابل نشان دهنده اولویت عملگرها در زبان C می‌باشد.

Token
() [] -> .
-- ++
-- ++ ! ~ - + & * Sizeof
(type)
* / %
+ -
<< >>
>>= <<=
= = !=
&
^
|
&&
||
?:
= += -= *= %= <<= >>= &= ^= |=
,

روشهای نمایش یک عبارت ریاضی 2+3*5

- روش میان ترتیب (Infix)
  - 2,+,3,*,5
- روش پیش ترتیب (Prefix)
  - +,2,*,3,5
- روش پس ترتیب (Postfix)
  - 2,3,5,*,+

روش infix

روش استاندارد نوشتن عبارات به عنوان infix معرفی و شناخته می شود چرا که در این روش عملگرهای دودویی را در بین دو عملوند قرار می دهیم.

نشانه گذاری postfix

دراین روش هر عملگر بعد از عملوند های مربوطه ظاهر می شود

مثال

Infix	Postfix
2+3*4	2 3 4*+
a*b+5	ab*5+
(1+2)*7	1 2+7*
a*b/c	ab*c/
((a/(b-c+d))*(e-a)*c)	abc-d+/ea-*c*
a/b-c+d*e-a*c	ab/c-de*+ac*-

نحوه محاسبه یک عبارت Postfix

عبارات برای ارزیابی از چپ به راست پویش می شوند. عملوندها تا مشاهده یک عملگر، داخل پشته قرار می گیرند، سپس تعداد لازم از عملوندها را از پشته خارج و پس از انجام عملکرد مربوطه ، نتیجه را دوباره به داخل پشته منتقل می کنیم . این کار را ادامه پیدا می کند تا به انتهای عبارت برسیم.

الگوریتم اول تبدیل infix به postfix

می توان الگوریتمی برای تبدیل یک عبارت infix به postfix به صورت زیر بیان نمود :

۱- پرانتزگذاری کامل عبارات

۲- انتقال همه عملگرهای دودویی به نحوی که با پرانتز بسته مربوطه سمت راست آن تعویض شوند

۳- حذف تمام پرانتزها

الگوریتم دوم تبدیل infix به postfix

- با استفاده از Stack میتوان عبارت infix را به postfix تبدیل نمود.

مثال

تبدیل عبارت a+b*c به نشانه گذاری postfix

Token	Stack [0] [1] [2]	Top	Output
a		-1	a
+	+	0	a
b	+	0	ab
*	+ *	1	ab
c	+ *	1	abc
eos		-1	abc*+

مثال

تبدیل a*(b+c)*d به نشانه گذاری postfix

Token	Stack [0]	[1]	[2]	Top	Output
a				-1	a
*	*			0	a
(	*	(		1	a
b	*	(		1	ab
+	*	(	+	2	ab
c	*	(	+	2	abc
)	*			0	abc+
*	*			0	abc+*
d	*			0	abc+*d
eos	*			0	abc+*d*