صف:(QUEUE)

تعریف: لیست مرتب شده ای است که عمل درج از یک سمت که rear نامیده میشود و عمل حذف از سمت دیگر آن که front نامیده می شود، انجام می شود.

به صف یا ساختمان داده FIFO (first in first out) گویند، یعنی اولین عنصری که وارد صف شد، اولین عنصری است که از صف خارج می شود. برای یک مثال از صف می توان به ماشینهای در حال انتظار در پمپ بنزین اشار کرد.

عملیاتی که روی صف می توان انجام داد عبارتند از:

initQ	عملیات آماده سازی صف را انجام می دهد.
addQ	درج یک عنصر در صف.
deleteQ	حذف یک عنصر از صف.
isEmpty	بررسی خالی بودن صف. صف وقتی خالی است که front و rear برابر باشند.
isFull	بررسی پر بودن صف. این بررسی بستگی به نوع پیاده سازی صف و نوع صف دارد. در صف معمولی که به کمک آرایه پیاده سازی می شود صف وقتی پر است که rear=MAX باشد و MAX تعداد عناصر آرایه است.

از کاربردهای صف می توان به مدیریت فرآیند ها در سیستم عامل اشاره نمود.

انواع صف:

صف خطی

صف حلقوی

صف اولویت

پیاده سازی صف

برای پیاده سازی صف هم میتوان از آرایه استفاده نمود و هم از لیست پیوندی. در اینجا به نحوه پیاده سازی صف به کمک آرایه اشاره می شود.

پیاده سازی صف در زبان C:

در پیاده سازی زیر از برای ذخیره عناصر صف از یک آرایه با گنجایش MAX_QUEUE_SIZE استفاده شده است.

```
#define  MAX_QUEUE_SIZE 100

class  Queue{
        private:
                int front;
                int rear;
                int elements[MAX_QUEUE_SIZE]
        public:
              Queue() { init() };
              void        Init();
              int         isEmpty();
              int         isFull();
              void  addQ( int data );
              int         deleteQ();
} Queue;


void          Queue::Init()
{
   front=rear=0;
}

int           Queue::isEmpty()
{
 return  (front==rear);
}

int           Queue::isFull()
{
   return (rear==MAX_QUEUE_SIZE);
}
```

```
void          Queue::addQ( int data )
{
   if (isFull()) {
         printf("ERROR: Queue is Full");
         exit(1);
   }
   elements[rear]=data;
   rear++;
}
int           Queue::deleteQ()
{
   if (isEmpty())
   {
         printf("ERROR: Queue is Empty");
         exit(1);
   }
   int  temp= elements[front];
   front++;
   return temp;
}
```

در پیاده سازی فوق، همیشه rear محلی را نشان که عنصر جدید باید در آنجا قرار بگیرد.

صف حلقوی :

مشکل صف خطی این است که با اضاف شدن عناصر به آن زمانی می رسد که صف پر میشود بنابراین دیگر نمی توان عنصر جدیدی را در آن درج کرد. و با خارج شدن عناصر از صف زمانی می رسد که هیچ عنصری در صف نیست یعنی front با rear برابر می شود در این زمان دیگر صف قابل استفاده نخواهد بود. برای حل این مشکل صف حلقوی ارائه می شود.

پیاده سازی صف حلقوی:

```
#define  MAX_QUEUE_SIZE 100

class  CQueue{
         private:
                  int front;
                  int rear;
```

```
                int elements[MAX_QUEUE_SIZE]
        public:
                Queue() { init() };
                void        Init();
                int         isEmpty();
                int         isFull();
                void  addQ( int data );
                int         deleteQ();
} Queue;
void            CQueue::Init()
{
   front=rear=0;
}
```

وضعیت خالی بودن صف حلقوی دقیقا مشابه صف خطی است که به صورت زیر بیان می ود.

```
int             CQueue::isEmpty()
{
 return  (front==rear);
}
```

وضعیت پر بودن صف حلقوی کمی متفاوت است. صف حلقوی زمانی پر است که rear دقیقا قبل از front باشد. در شکل زیر وضعیت پربودن یک صف حلقوی با گنجایش ۹ عنصر نشان داده شده است.

		rear	front					
14	13		10	15	441	55	69	90

یک وضعیت دیگر ممکن است به صورت زیر باشد

front								Rear
10	12	14	189	52	26	144	22	

با توجه به مطالب فوق می توان تابع isFull را مطابق کد زیر پیاده سازی نمود.

```
int            CQueue::isFull()
{
   return ((rear+1)% MAX_QUEUE_SIZE== front);
}
```

تفاوت توابع درج و حذف در لیست حلقوی و لیست خطی در مقدار دهی rear و front می باشد.

به طوری که مقدار جدید آنها از فرمول زیر محاسبه میشود.

```
rear= (rear+1) % MAX_QUEUE_SIZE;
front=(front+1) % MAX_QUEUE_SIZE;
```

برای درک بهتر می توانید توابع درج و حذف در لیست حلقوی را مشاهده نمایید.

```
void           CQueue::addQ( int data )
{
   if (isFull()) {
         printf("ERROR: Queue is Full");
         exit(1);
   }
   elements[rear]=data;
   rear=(rear+1) % MAX_QUEUE_SIZE;
}

int            CQueue::deleteQ()
{
   if (isEmpty())
   {
         printf("ERROR: Queue is Empty");
         exit(1);
   }
   int  temp= elements[front];
   front=(front+1) % MAX_QUEUE_SIZE;
   return temp;
}
```