داده ساختارها و الگوریتم ها

Data Structures

میلاد سلطانی

فهرست مطالب فصل سوم

- مقدمه
- تعریف صف
- صف ساده و حلقوی
- عملیات روی صف ساده و حلقوی
  - درج
  - حذف
  - بازیابی
- مشکلات صف‌ها
- کاربرد صف

صف‌ها

Queue

مقدمه

- در فرهنگ لغات صف به معنای " خط انتظار" است.
- در هر نوع صف، مثل صف نانوایی، صف ماشین ها جلوی عوارضی بزرگراه و ... افراد یا اشیاء به ترتیب که وارد می شوند، خدمات دریافت می کنند.
- صف مجموعه ای از عناصر مرتب است که هر عنصر آن، از طرف جلوی صف front حذف می شود و از طرف آخر صف rear در آن قرار می گیرد.
- عملکرد ساختمان داده صف " اولین ورودی، اولین خروجی " می باشد که به اختصار FIFO گفته می شود.

صف به عنوان یک ADT

- صف را می توان بصورت یک نوع داده انتزاعی در نظر گرفت :

ADT صف

عناصر داده:

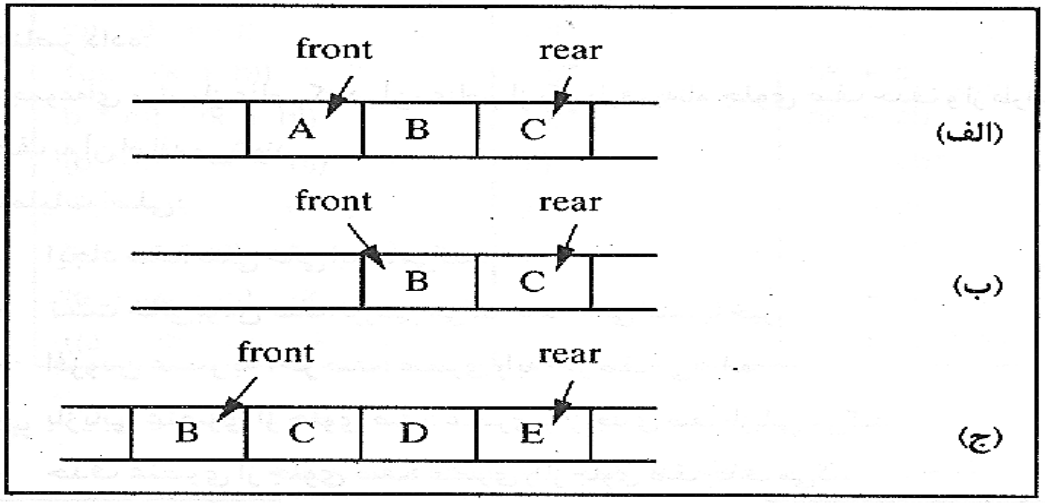
مجموعه‌ای مرتب از عناصر که در آن، عناصر از یک طرف به‌نام **جلوی صف** حذف و از طرف دیگر به‌نام **آخر صف** به آن اضافه می‌شوند.

عملیات اصلی:

- **ایجاد صف:** صفی خالی ایجاد می‌کند.
- **تست خالی‌بودن صف:** بررسی می‌کند صف خالی است یا خیر.
- **افزودن عنصر به آخر صف:** عنصری را به آخر صف می‌افزاید.
- **بازیابی عنصری از جلوی صف:** عنصری را از جلوی صف بازیابی می‌کند.
- **حذف عنصری از جلوی صف:** عنصری را از جلوی صف حذف می‌کند.

پیاده سازی صف

- مثل پشته، صف را می توان به دو صورت نمایش داد :
  ۱. خطی
  ۲. پیوندی
- در روش خطی از آرایه برای نگهداری عناصر صف استفاده می شود و در روش پیوندی، لیست های پیوندی استفاده می شوند.



نمونه‌ای از عملیات بر روی صف

پیاده سازی صف با آرایه

- برا نمایش صف با آرایه به موارد زیر نیاز است:
  - آرایه ای برای خیره عناصر صف
  - متغیر نشان دهنده جلوی صف
  - متغیر نشان دهنده انتهای صف

```
#define SIZE 100
struct   queue {
         int items[SIZE];
         int front;
         int rear;
};
queue q;
q.front = 0;
q.rear = -1;
```

نکته ! بر اساس این پیاده سازی، تعداد عناصر صف در هر لحظه برابر است با : $q.rear - q.front + 1$

پیاده سازی تست پر بودن صف

- قبل از درج عنصر در یک صف بایستی اطمینان حاصل کرد که صف پر نیست.
- بر اساس پیاده سازی خطی صف، زمانی یک صف پر فرض می شود که :

$$q.rear = SIZE - 1$$

```
int  full (queue *q)
{
    if(q ->rear == SIZE-1)
        return 1;
    return 0;
}
```

آیا صف پر است ؟

بله صف پر است.

خیر صف پر نیست.

نکته ! پر نبودن صف به معنای خالی بودن آن نیست بلکه به این معناست که حداقل یک فضای خالی در صف وجود دارد.

پیاده سازی عمل درج در صف

- پس از اطمینان از پر نبودن صف، برای درج عنصر باید :
  ۱) متغیر انتهای صف یک واحد افزایش می یابد.
  ۲) عنصر به انتهای صف اضافه می شود.

```
void addQ(queue *q, int x, int *overflow)
{
   if(full(q))
      *overflow = 1;
   else
   {
      *overflow = 0;
      q->items[++(q -> rear)] = x;
   }
}
```

تست پر بودن صف

اگر صف پر نیست

عنصر به انتهای صف اضافه می شود.

نکته ! در این تابع x عنصری است که اضافه می شود. overflow اگر 1 شود یعنی صف پر بوده و اگر 0 شود یعنی صف پر نیست.

پیاده سازی تست خالی بودن صف

- قبل از حذف عنصر از یک صف بایستی اطمینان حاصل کرد که صف خالی نیست.
- بر اساس پیاده سازی خطی صف، زمانی یک صف خالی فرض می شود که :

$$q.rear < q.front$$

```
int  empty(queue *q)
{
    if(q -> rear < q -> front)
        return 1;
    return 0;
}
```

آیا صف خالی است ؟

بله صف خالی است.

خیر صف خالی نیست.

نکته ! خالی نبودن صف به معنای پر بودن آن نیست بلکه به این معناست که حداقل یک عنصر در صف وجود دارد.

پیاده سازی عمل حذف از صف

- پس از اطمینان از خالی نبودن صف، برای حذف عنصر باید :
  ۱) عنصر سر صف برداشت می شود.
  ۲) متغیر جلوی صف یک واحد اضافه می شود.

```
void qremove(queue *q, int *x, int *underflow)
{
   if(empty(q))
      *underflow = 1;
   else
   {
      *underflow = 0;
      *x = q -> items[(q -> front) ++];
   }
}
```

تست خالی بودن صف

اگر صف خالی نیست

عنصر سر صف حذف می شود.

نکته ! در این تابع x عنصر حذف شده را نگهداری می کند و underflow اگر 1 شود یعنی صف خالی بوده و اگر 0 شود یعنی صف خالی نیست.

پیاده سازی عمل بازیابی از صف

- عمل بازیابی کاملاً مشابه عمل حذف است ولی با این تفاوت که در بازیابی عنصری حذف نمی شود. فقط عنصر سر صف نمایش داده می شود.

```
void retrieve(queue *q, int *x, int *underflow)
{
   if(empty(q))
        *underflow = 1;
   else
   {
        *underflow = 0;
        *x = q -> items[q -> front];
   }
}
```

تست خالی بودن صف

اگر صف خالی نیست

عنصر سر صف قابل دسترسی است.

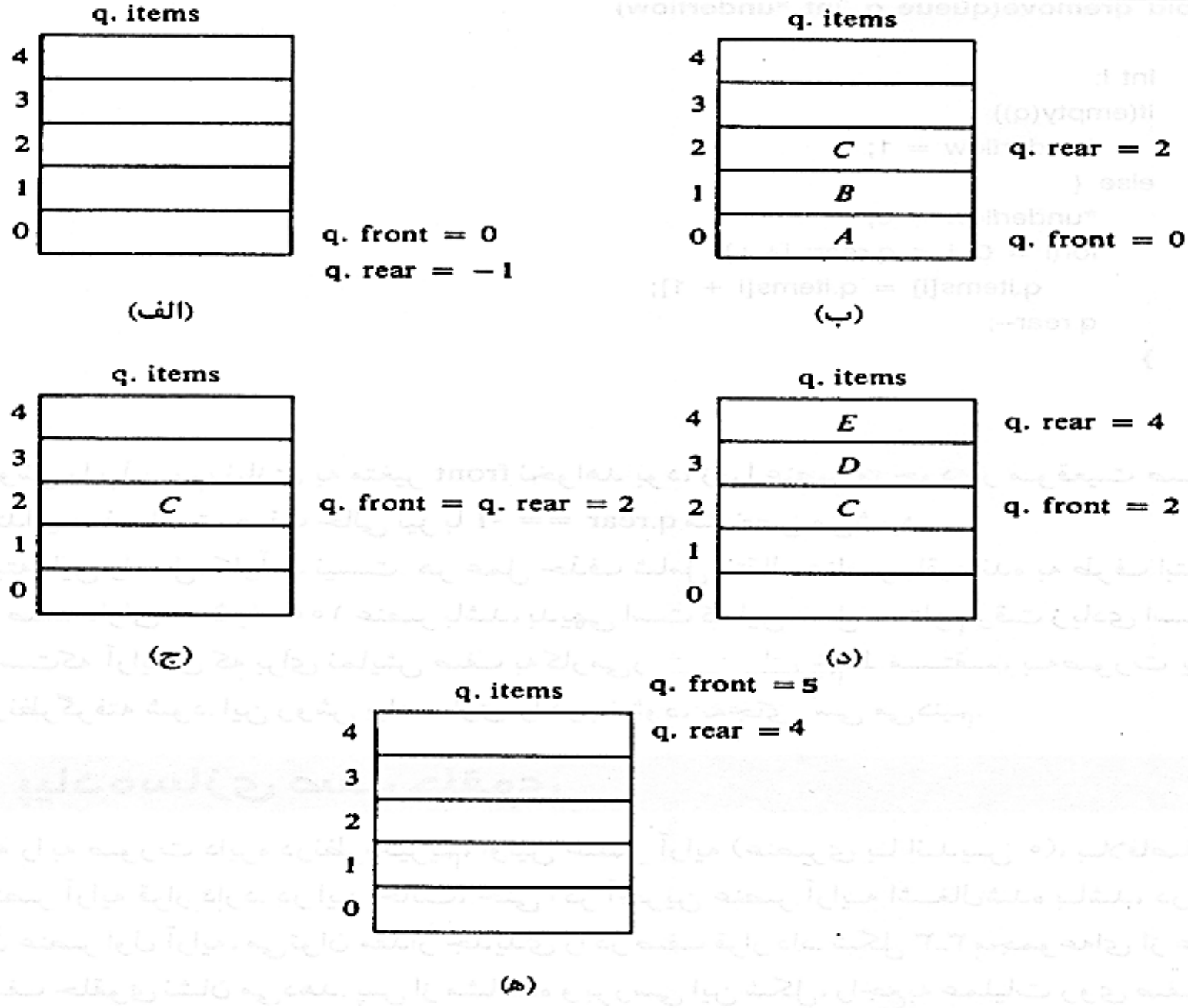
مشکلات پیاده سازی صف با آرایه

- مشکلات صف پیاده سازی شده با آرایه به دو دسته تقسیم می شود :
  ۱. زمانیکه صف جا دارد نمی توان در آن عنصری درج کرد.
  ۲. زمانیکه صف کاملاً خالی است نمی توان عنصری در آن درج کرد.

مثال : در شکل صفحه بعد یک صف مشاهده می کنید. عملیات زیر را به ترتیب از راست به چپ روی آن انجام دهید.

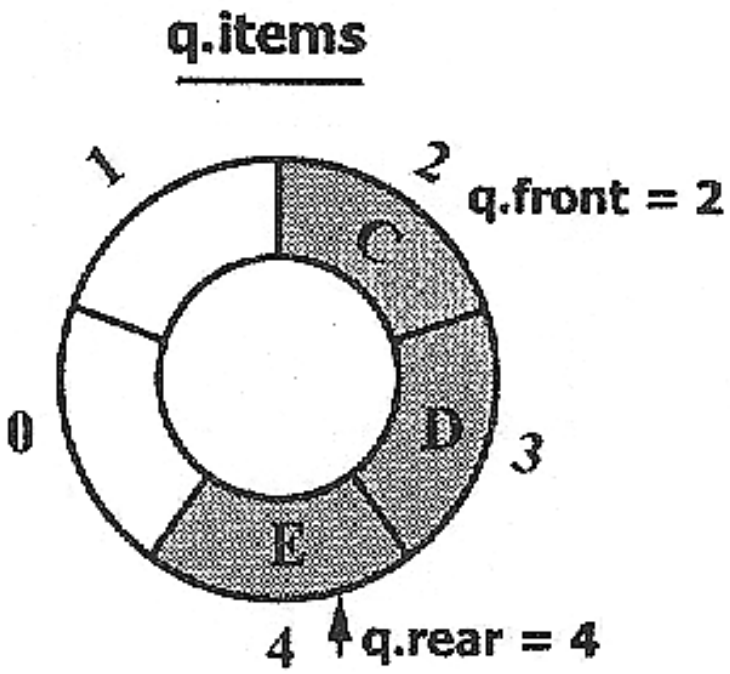
(a درج A – درج B – درج C – حذف عنصر- حذف عنصر- درج D – درج E – درج F – چه مشکلی در این عملیات وجود دارد ؟

(b در ادامه عملیات قبلی : حذف عنصر- حذف عنصر- حذف عنصر- درج F – چه مشکلی در این عملیا وجود دارد ؟


q. items
4
3
2
1
0
q. front = 0
q. rear = −1
(الف)
q. items
4
3
2
C
1
B
0
A
q. rear = 2
q. front = 0
(ب)
q. items
4
3
2
C
1
0
q. front = q. rear = 2
(ج)
q. items
4
E
3
D
2
C
1
0
q. rear = 4
q. front = 2
(د)
q. items
4
3
2
1
0
q. front = 5
q. rear = 4
(هـ)

پیاده سازی صف حلقوی

- برای جلوگـیری از مشکلات ذکر شده، صف خطی با آرایـش حلقـوی ایجاد می شود. یعنی انتهای صف به ابتدای صف متصل می شود.

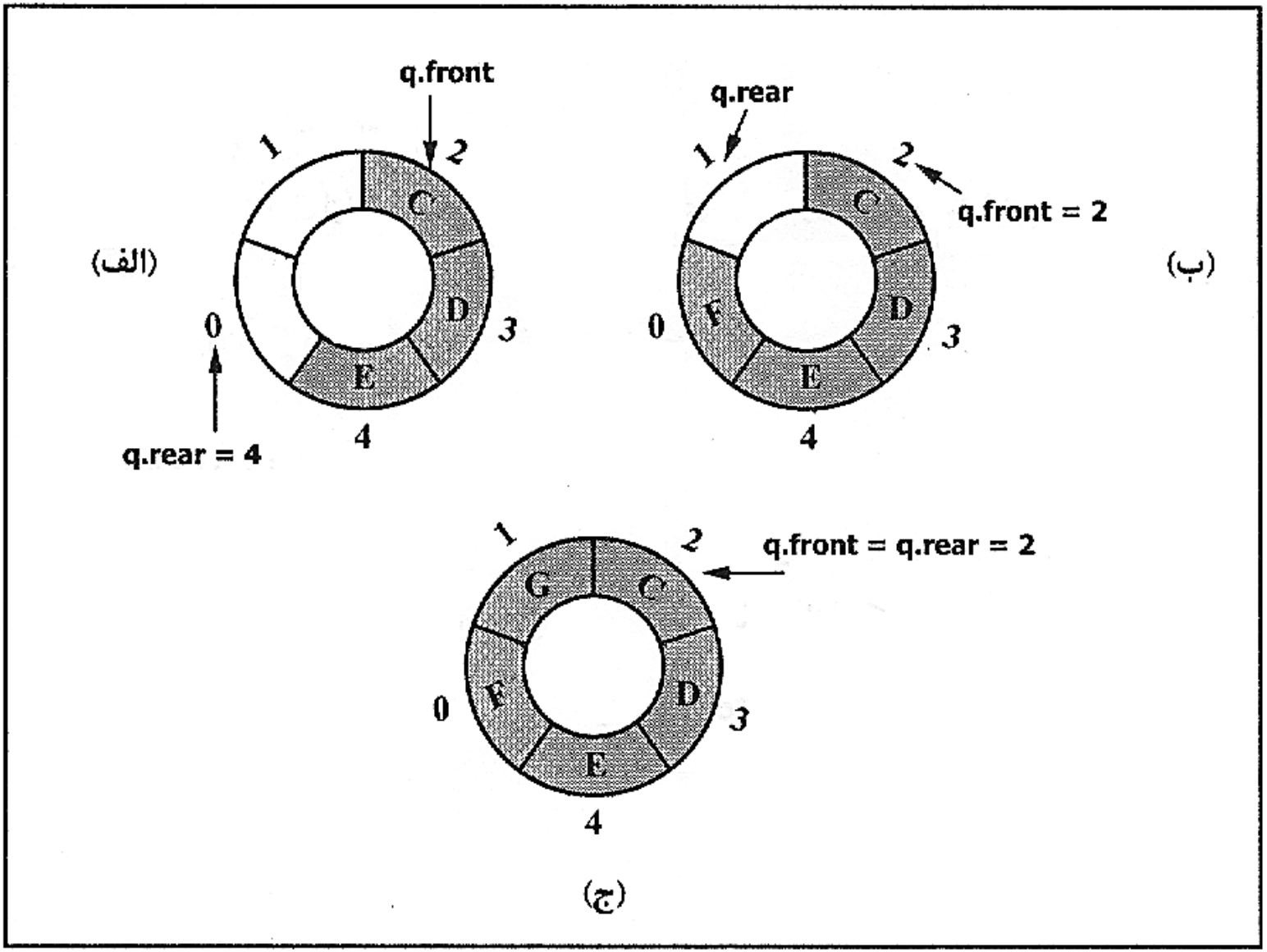


پیاده سازی صف حلقوی

- برا نمایش صف حلقوی با آرایه به موارد زیر نیاز است:
  - آرایه ای برای ذخیره عناصر صف
  - متغیر نشان دهنده جلوی صف
  - متغیر نشان دهنده انتهای صف

```
#define SIZE 100
struct   queue {
         int items[SIZE];
         int front;
         int rear;
};
queue q;
q.rear = q.front = 0;
```

نکته ! در این پیاده سازی صف طول صف همیشه یک واحد از اندازه آرایه آن کمتر فرض می شود تا خالی و پر بودن صف قابل تشخیص باشند.


q.front
1
2
C
D
3
0
E
4
q.rear = 4
(الف)
q.rear
1
2
C
q.front = 2
D
0
F
3
E
4
(ب)
1
2
q.front = q.rear = 2
G
C
0
F
D
3
E
4
(ج)

پیاده سازی تست پر بودن صف حلقوی

- قبل از درج عنصر در یک صف بایستی اطمینان حاصل کرد که صف پر نیست.
- بر اساس پیاده سازی خطی صف حلقوی، زمانی یک صف پر است که :

$$(q.rear + 1) \% SIZE = q.front$$

```
int full (queue q)
{
    if((q->front) = ((q -> rear) + 1) % SIZE;)
        return 1;
    return 0;
}
```

آیا صف پر است ؟

بله صف پر است.

خیر صف پر نیست.

نکته ! پر نبودن صف به معنای خالی بودن آن نیست بلکه به این معناست که در صف حداقل یک فضای خالی وجود دارد.

پیاده سازی عمل درج در صف حلقوی

- پس از اطمینان از پر نبودن صف، برای درج عنصر باید :
  ۱) عنصر به انتهای صف اضافه می شود.
  ۲) متغیر انتهای صف یک واحد افزایش می یابد.

```
void addQ(queue *q, int x, int *overflow)
{
   int newRear;
   newRear = ((q -> rear) + 1) % SIZE;
   if(full(q))
       *overflow = 1;
   else
   {
      *overflow = 0;
      q -> items[q -> rear] = x;
      q -> rear = newRear;
   }
}
```

تست پر بودن صف

اگر صف پر نیست

عنصر به انتهای صف اضافه می شود.

نکته ! در این تابع x عنصری است که اضافه می شود. overflow اگر 1 شود یعنی صف پر بوده و اگر 0 شود یعنی صف پر نیست.

پیاده سازی تست خالی بودن صف حلقوی

- قبل از حذف عنصر از یک صف بایستی اطمینان حاصل کرد که صف خالی نیست.
- بر اساس پیاده سازی خطی صف حلقوی، زمانی یک صف خالی فرض می شود که :

q.rear = q.front

```
int empty(queue q)
{
    if(q.rear == q.front)
        return 1;
    return 0;
}
```

آیا صف خالی است ؟

بله صف خالی است.

خیر صف خالی نیست.

نکته ! خالی نبودن صف به معنای پر بودن آن نیست بلکه به این معناست که حداقل یک عنصر در صف وجود دارد.

پیاده سازی عمل حذف از صف حلقوی

- پس از اطمینان از خالی نبودن صف، برای حذف عنصر باید :
  ۱) عنصر سر صف برداشت می شود.
  ۲) متغیر جلوی صف یک واحد اضافه می شود.

```
void qremove(queue *q, int *x, int *underflow)
{
   if(empty(q))
       *underflow = 1;
   else
   {
       *underflow = 0;
       *x = q -> items[(q -> front)];
       q -> front = ((q -> front) + 1) % SIZE;
   }
}
```

تست خالی بودن صف

اگر صف خالی نیست

عنصر سر صف حذف می شود.

نکته ! در این تابع x عنصر حذف شده را نگهداری می کند و underflow اگر 1 شود یعنی صف خالی بوده و اگر 0 شود یعنی صف خالی نیست.

پیاده سازی عمل بازیابی از صف حلقوی

- عمل بازیابی کاملاً مشابه عمل حذف است ولی با این تفاوت که در بازیابی عنصری حذف نمی شود. فقط عنصر سر صف نمایش داده می شود.

```
void retrieve(queue *q, int *x, int *underflow)
{
   if(empty(q))
       *underflow = 1;
   else
   {
       *underflow = 0;
       *x = q -> items[q -> front];
   }
}
```

تست خالی بودن صف

اگر صف خالی نیست

عنصر سر صف قابل دسترسی است.

نکات صف حلقوی

- در هنگام اصلاح متغیر انتهای صف حلقوی حین عمل درج کردن رابطه زیر را می نویسیم :

$$q.rear = (q.rear + 1) \% SIZE$$

- در هنگام اصلاح متغیر جلوی صف حلقوی حین عمل حذف کردن رابطه زیر را می نویسیم :

$$q.front = (q.front + 1) \% SIZE$$

- شرط خالی بودن صف حلقوی : $q.rear = q.front$

- شرط پر بودن صف حلقوی : $(q.rear + 1) \% SIZE = q.front$

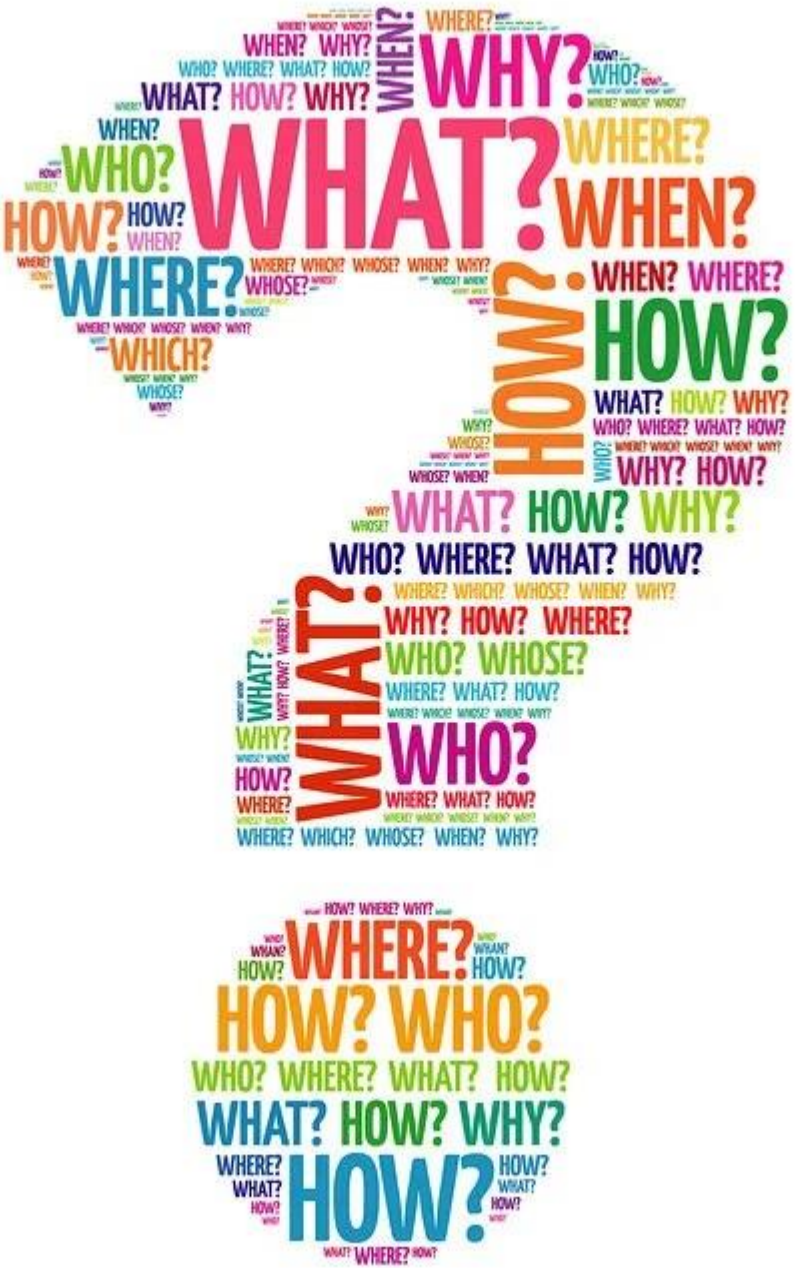
کاربردهای صف

- از کاربرد های صف می توان به موارد زیر اشاره کرد :

۱) زمان بندی پردازنده در محیط چند برنامه ای

۲) الگوریتم نوبت گردشی در سیستم عامل

- تمام این کاربردها در درس سیستم عامل مورد بررسی قرار خواهد گرفت.


WHERE?
WHEN? WHY?
WHEN?
WHY?
WHO? WHERE? WHAT? HOW?
WHAT? HOW? WHY?
WHEN?
WHO?
WHAT?
WHERE?
HOW?
HOW?
WHEN?
WHERE?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHOSE?
WHEN? WHERE?
HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHICH?
WHOSE?
WHAT? HOW? WHY?
WHO? WHERE? WHAT? HOW?
WHY? HOW?
WHAT? HOW? WHY?
WHO? WHERE? WHAT? HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHY? HOW? WHERE?
WHO? WHOSE?
WHERE? WHAT? HOW?
WHAT?
WHY?
HOW?
WHERE?
WHO?
WHERE? WHAT? HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
HOW? WHERE? WHY?
WHERE?
HOW?
HOW? WHO?
WHO? WHERE? WHAT? HOW?
WHAT? HOW? WHY?
WHERE?
WHAT?
HOW?
HOW?
HOW?
WHAT?
HOW?
WHAT? WHERE? HOW?