Filas (Colas, Queue)

Estructuras de Datos

1

Fifo: First In First Out

Las filas son un subconjunto de las listas, en donde las eliminaciones se dan al comienzo de la lista y las inserciones al final.

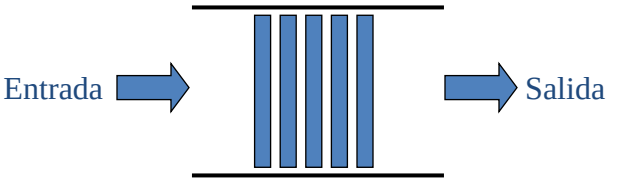
Su protocolo E/S es **FIFO:**

First In, First Out.

2

FILA (Conceptos ....)

- Es un TDA que, sobre una lista de elementos, utiliza el protocolo E/S **FIFO** (First In, First Out)



3

LA FILA (Queue)

- Se trabaja con dos puntos de acceso, la cabeza (inicio, front) y el final (fin, rear).

- Entre sus operaciones se encuentran las de agregar un elemento a la fila (enqueue) y la de eliminar un elemento de la fila (dequeue).

4

FILA

- Cuando se **agrega** un elemento, se coloca al **final** de la cola, cuando se **elimina**, se elimina del **inicio** de la misma.

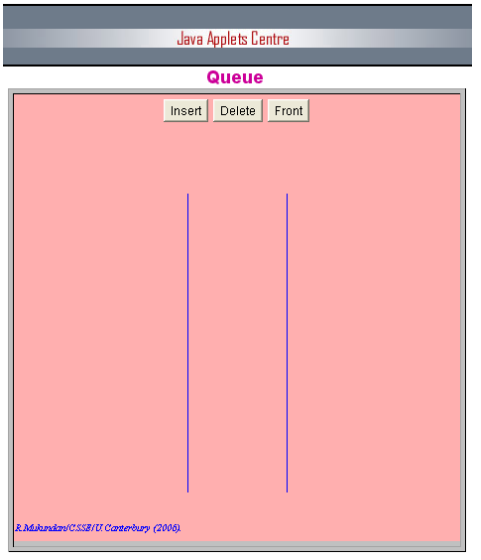
5

Primero en entrar, primero en salir…

6

Funcionamiento de una Fila

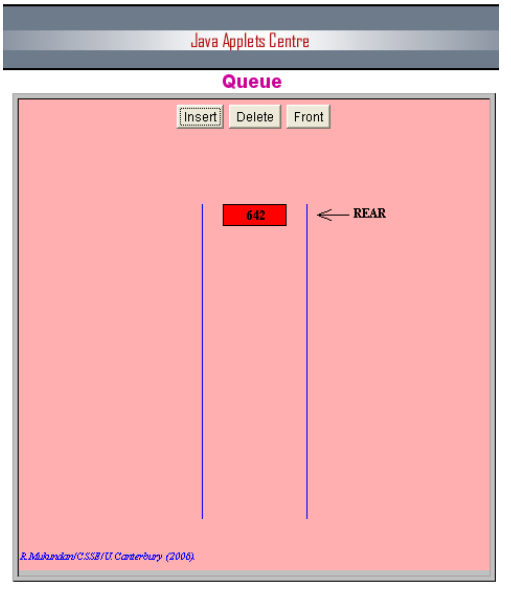
Cola vacía



7

Funcionamiento de una Fila

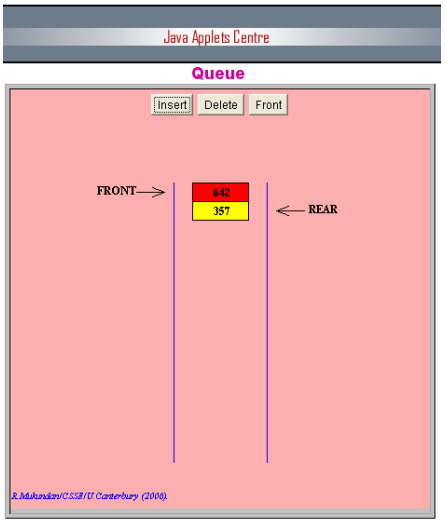
Inserta 642



8

Funcionamiento de una Fila

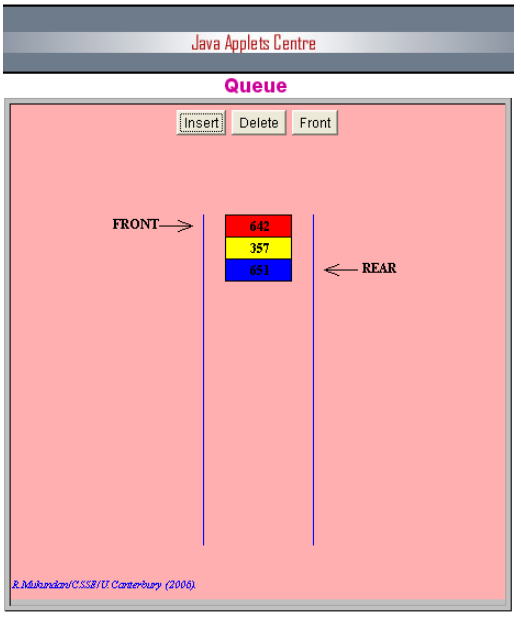
Inserta 357



9

Funcionamiento de una Fila

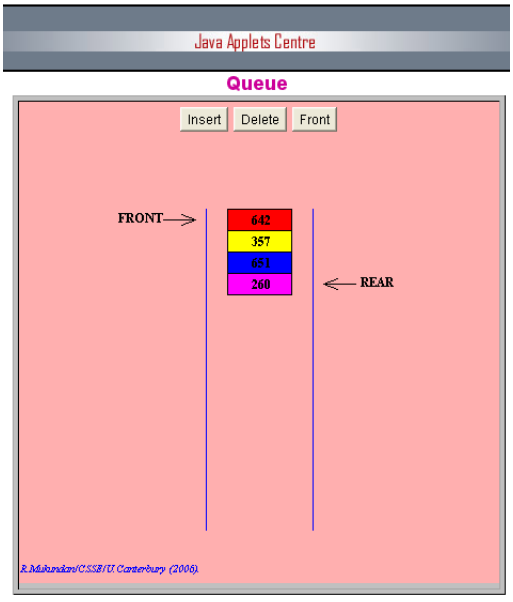
Inserta 651



10

Funcionamiento de una Fila

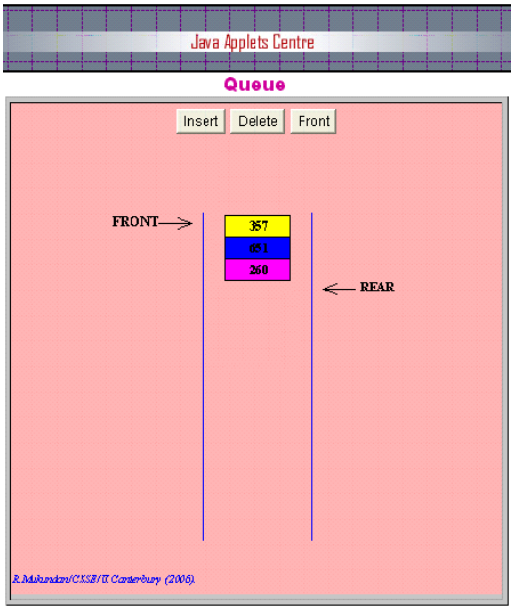
Inserta 260



11

Funcionamiento de una Fila

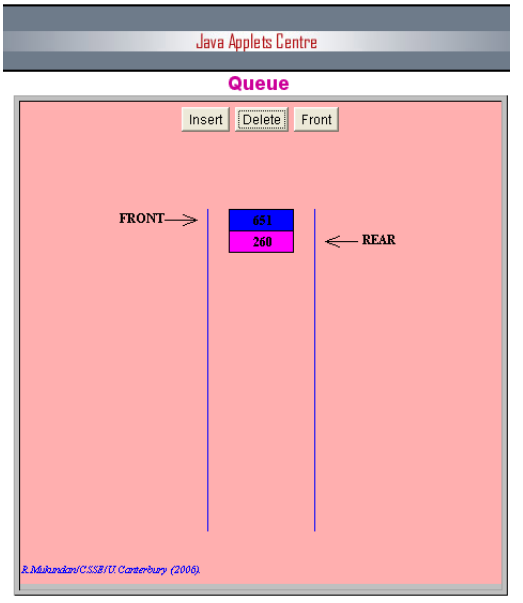
Retira un elemento….



12

Funcionamiento de una Fila

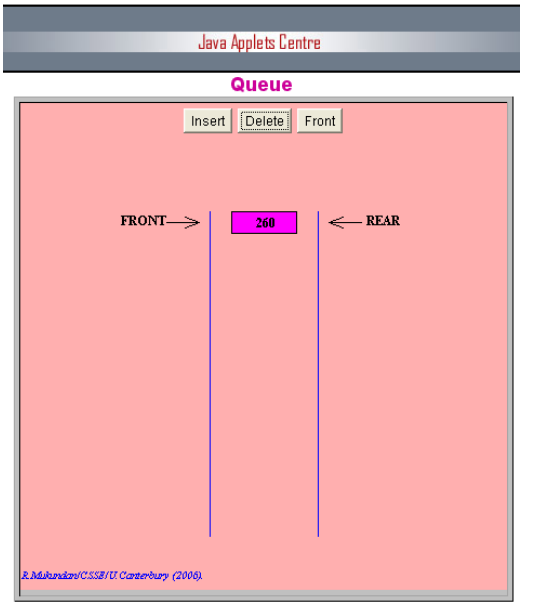
Delete



13

Funcionamiento de una Fila

Delete



14

Estructura Nodo Fila

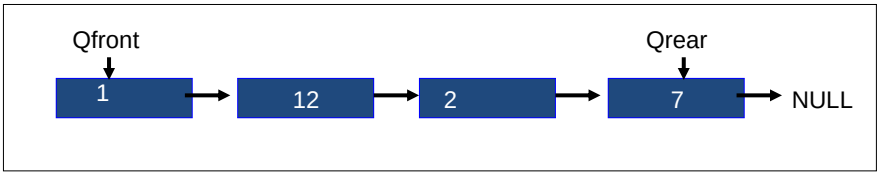
```
struct fila
{
  int dato;
  struct fila *sig;
};

typedef struct fila NODO;
```

15

Implementación Dinámica

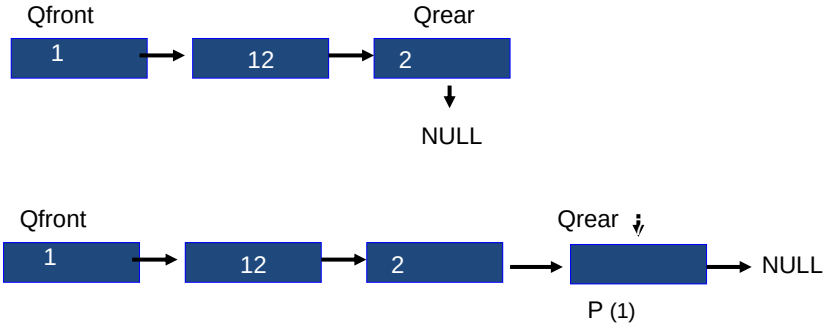
- Qfront y Qrear son punteros al inicio y al final de la fila, respectivamente:



16

Inserción en una Fila

- La inserción se produce siempre al fondo de la fila (Qrear):



17

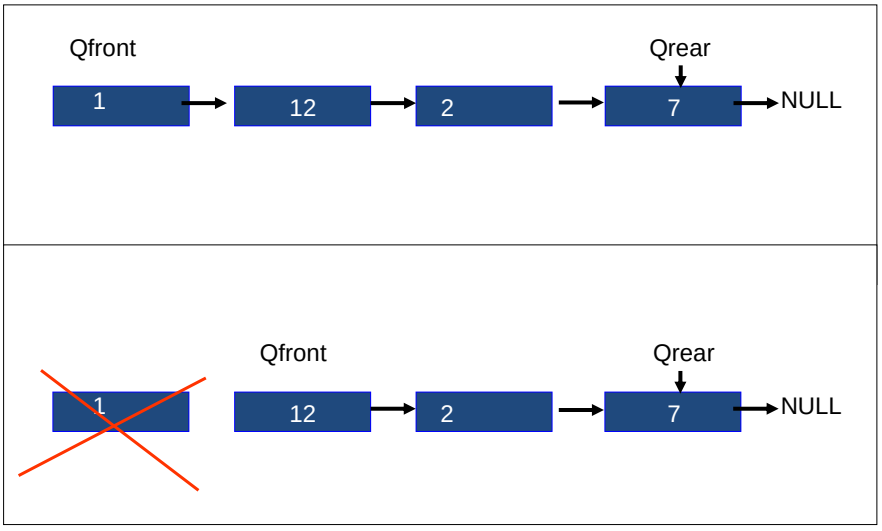
Inserción de un elemento

```
void inqueue(NODO **qrear, NODO **qfront, int N)
{
 NODO *p=NULL;
 p=new(NODO);
 p->dato=N;
 p->sig=NULL;
 if(*qrear==NULL){
              *qfront=p;
              *qrear=p;
              }
 else{
      qrear ->sig=*p;
     *qrear=p;
     }
}
```

18

Eliminación en una Fila

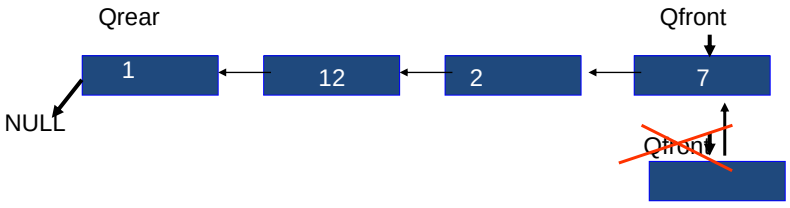
La eliminación se produce siempre al frente de la fila (Qfront):



19

Eliminación en una fila

- La eliminación siempre es al principio de la fila (FIFO):



20

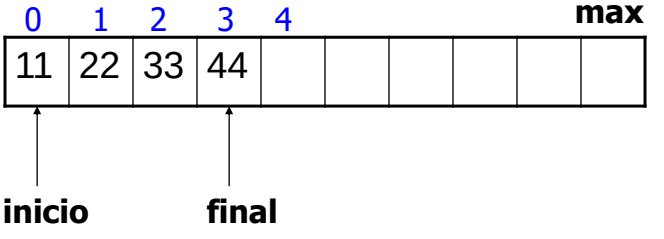
Eliminación de un nodo en una Fila

```
void dequeue(NODO **qrear, NODO **qfront)
{
   NODO *p=NULL,*q=NULL;
   if (*qfront== NULL){
         printf("\n Fila vacía....");}
   else
       if (*qfront==*qrear){p=*qfront;
                            *qrear=NULL;
                           *qfront=NULL;
                            delete(p);
                   }
       else{
           p=*qfront;
          *qfront= (*qfront)->sig;
           delete(p);
           }
}
```

21

Fila con estructura estática

Es una fila implementada como un arreglo



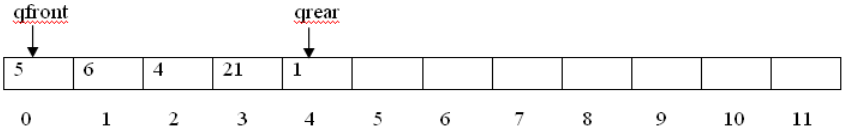
22

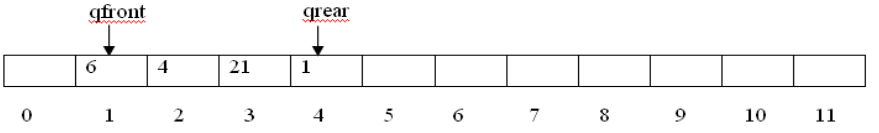
Filas: **Fila vacía**

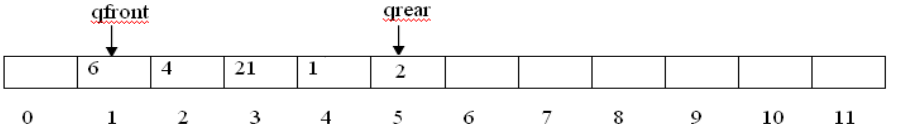
final
inicio

23

Implementación Estática

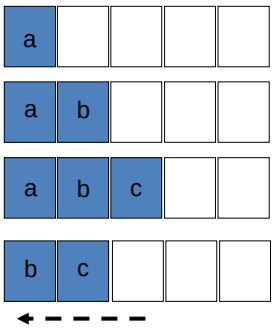




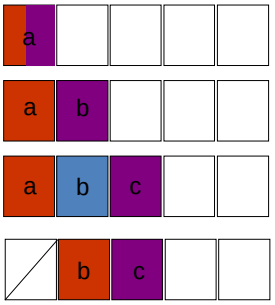


24

Problemas con la implementación estática:



Es necesario mover todos los elementos, con el esfuerzo que ello significa



O asumir que irá quedando espacio desperdiciado al frente de la cola (mala opción)

25

Implementación con arrays

- Existe un problema en la utilización eficiente del espacio de memoria asignado, llega un momento en el que se desborda la capacidad del array.
- Una solución poco efectiva es incrementar su tamaño. Esta implementación es sencilla pero totalmente ineficaz.
- La solución pasa por una implementación en un vector circular.

26

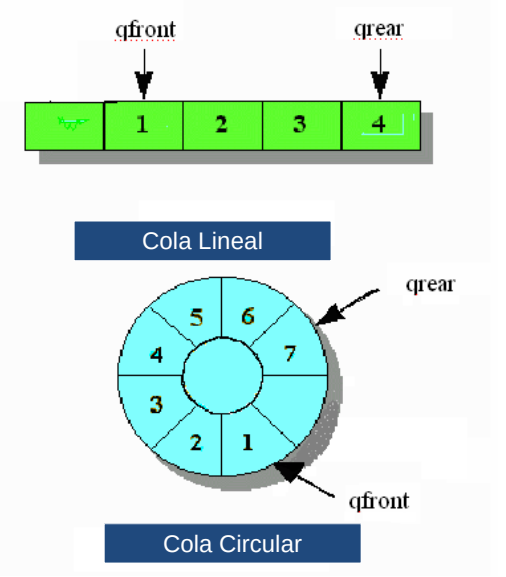
Filas con vectores circulares

- Se utilizará una variable para “contar” el número de elementos en el vector. Si el total de elementos es igual a la capacidad máxima del vector, la fila está llena.

- Qfront y qrear serán variables que almacenarán las posiciones de inicio y término en la fila circular.

27

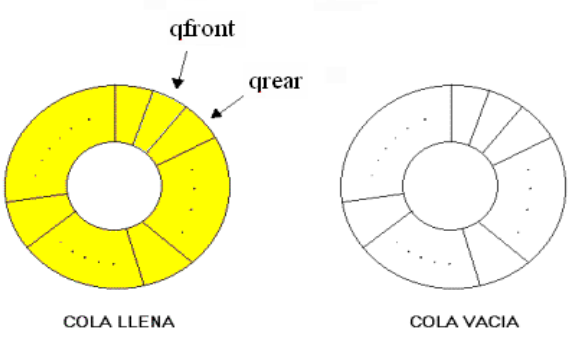
Implementación con vectores



28

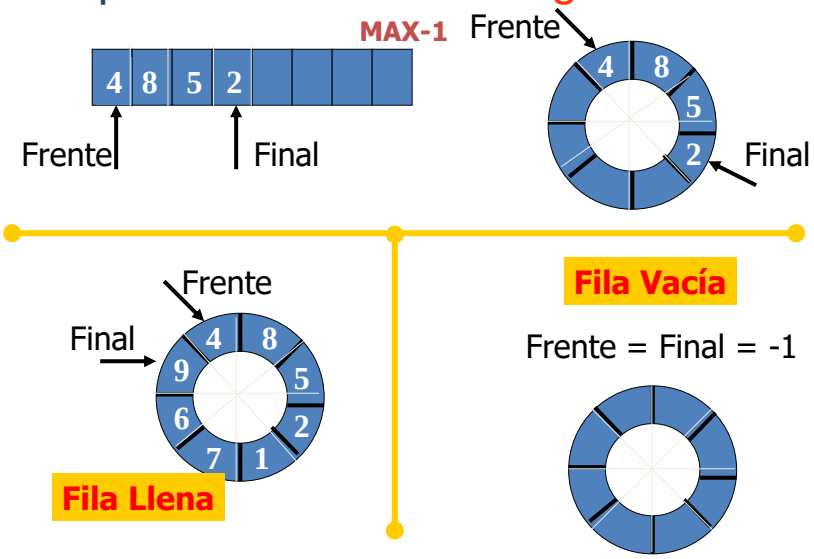
Fila con array circular

Fila vacía → qfront = qrear = -1 (no existe)



29

Implementación en Arreglos



30

```
// Creación de la Fila Estática

int newqueue(int &qfront, int &qrear, int &cont)
{   int var;
    qfront=qrear=0; cont++;
    printf("\n Ingrese el primer elemento de la
cola: ");
    scanf("%d",&var);
    return(var);
}
```

31

Inserción en Fila Estática Circular.

```
void inqueue(int &qfront,int &qrear, int
    &cont, int cola[])
{
if (cont < max) {
        if(qrear == max-1) qrear=0;
        else qrear++;
        cont++;
        printf("\nIngrese el siguiente elemento
    de la fila:");
       scanf("%d",&cola[qrear]);
        printf("\n\n Seguimiento: en inqueue,
    qfront %d, qrear %d, cont %d ", qfront,
    qrear, cont);
        }
else printf("\n overflow: la fila está llena...");
}
```

32

Elimina-ción en Fila Estática Circular.

```
int dequeue(int &qfront,int &qrear, int &cont, int
    cola[])
{int aux=qfront;
if (cont != 0) {
   if (qfront == max-1)qfront=0;
   else qfront++;
   cont--;
   printf("\n en dequeue, qfront %d, qrear %d,
    cont %d cola:", qfront, qrear, cont, cola[qrear]);
   return (cola[aux]);
}
else {printf("\Fila vacía, no puede retirar un
    elemento...");
    return(0);
    }
}
```

33

Impresión en Fila Estática Circular.

```
void printqueue(int qfront, int qrear, int cont,
    int cola[])
{
int i,j=0;
 if (cont == 0) printf("\n la cola está vacía...");
 else {
      printf("\n La cola contiene los siguientes
    elementos (en orden de llegada):");
     i=qfront;
     while (j<cont) {
                    printf("  %d", cola[i]); j++;
                    if(i==max-1)i=0;
                    else i++;
                    }
}
}
```

34

```
main()
{
int cola[max];
int N,c,i,qrear,qfront,cont=0;
:
:
```

35