TDA COLA

Ing. Mario Milton López Winnipeg

Capitulo 4 TDA Cola

4 Colas

4.1 Descripción del TDA Cola.

4.2 Especificación del TDA Cola.

4.3 Ejemplos de uso.

4.4 Implementaciones del TDA Cola.

- 4.4.1 Implementación basada en el TDA Lista.
- 4.4.2 Implementación con vectores.
- 4.4.3 Implementación con apuntadores.
- 4.4.4 Comparación de las implementaciones.

4.1 Descripción del TDA Cola

- Una cola es un caso particular de lista en el cual los elementos se insertan en un extremo (el posterior o final) y se suprimen en el otro (el anterior o frente).

- Las colas se conocen también como listas *FIFO* (*first-in first-out*) o listas ``primero en entrar, primero en salir''.

- Algunas de las operaciones vistas para listas pierden sentido en el TDA Cola y se definen nuevas operaciones

4.1 Descripción del TDA Cola

- Podemos definir las siguientes operaciones:
  - CREA. Crea una cola vacía.
  - VACIA. Devuelve un valor cierto si la cola está vacía, y falso en caso contrario.
  - PRIMERO. Devuelve el primer elemento de una cola.
  - PONER. Añade un elemento por el extremo final de una cola.
  - SACAR. Suprime el primer elemento de una cola y retorna dicho valor.

4.2 Especificación del TDA Cola

Especificación Informal

- `Cola` = **TDA con operaciones** `crea`, `vacia`, `primero`, `poner`, **sacar.**

- **DESCRIPCIÓN:**
  - **Los valores del TDA** `Cola` **son colas de elementos del tipo** `Elemento`**. Las colas son mutables:** `poner` **y sacar añaden y eliminan elementos en la cola respectivamente.**

- **OPERACIONES:**
  - `crea()` `devuelve (C:Cola)`
    - **efecto: Devuelve la cola vacía** `C`**.**

4.2 Especificación del TDA Cola

- `vacia(C:Cola) devuelve (booleano)`
  - efecto: Devuelve `cierto` si `C` es la cola vacía, y `falso` en caso contrario.
- `primero(C:Cola) devuelve (E:Elemento)`
  - requerimientos: La cola `C` es no vacía.
  - efecto: Devuelve en `E` el primer elemento de la cola `C`.
- `poner(C:Cola; E:Elemento)`
  - modifica: C.
  - efecto: Añade el elemento E por el extremo final de la cola C
- `sacar(C:Cola, ES E: Elemento)`
  - requerimientos: La cola C es no vacía.
  - modifica: C.
  - efecto: Suprime el primer elemento de la cola C y lo retorna.

4.2 Especificación del TDA Cola

Especificación Formal

Tipo: `Cola (Elemento)`

- *Sintaxis:*
  - `crea → Cola`
  - `vacia(Cola) → booleano`
  - `primero(Cola) → Elemento`
  - `poner(Cola,Elemento) → Cola`
  - `sacar(Cola, Elemento) → Cola`

4.2 Especificación del TDA Cola

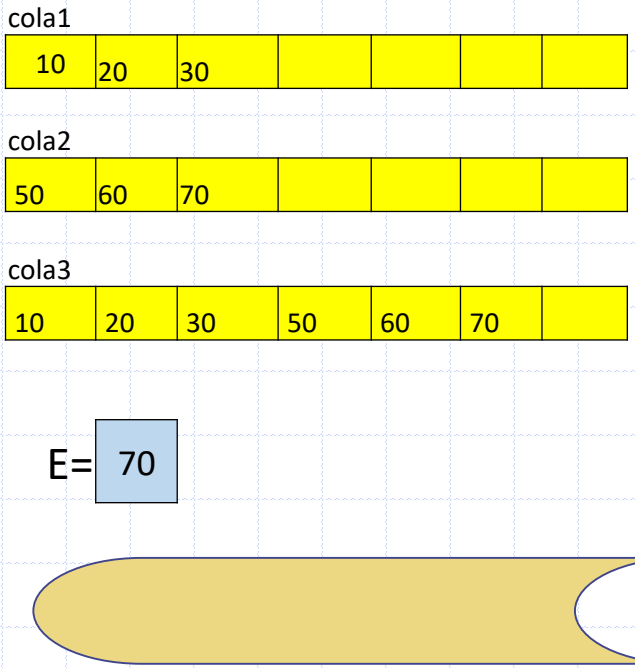
La interface del TDA Cola de acuerdo a esta especificación puede definirse de la siguiente forma:

```
publico interface Cola
{
     publico boolean vacia();
    publico Elemento Primero()
     publico poner(Objecto elemento);
     publico sacar(Objeto elemento);
}  // fin interface Cola
```

4.3 Ejemplo de uso

```
publico concatenar(Cola cola1, Cola cola2 ES Cola Cola3)
  inicio
      mientras No (cola1.vacia()= Verdadero)
      inicio
           cola1.sacar(E ) colaAux.Poner(E)
      fin
       mientras no(colaAux.vacia())
         inicio
              colaAux.Sacar(E);
              cola3.Poner(E);
              cola1.poner(E)
         fin

      mientras No (cola2.vacia()= Verdadero)
       inicio
           cola2.sacar(E ) colaAux.Poner(E)
       fin
      mientras no(colaAux.vacia())
       inicio
          colaAux.Sacar(E);
          cola3.Poner(E);
          cola2.poner(E)
      fin
  fin
```



4.4 Implementaciones del TDA Cola

- En esta sección mostraremos tres implementaciones para el TDA Cola:
  - Implementación basada en el TDA Lista
  - Implementación con vectores circulares
  - Implementación con apuntadores

4.4.1 Implementación basada en el TDA Lista

```
Clase Cola
    Atributos
        L : Lista
    Metodos
        Crear()
        Vacia()
        Poner(E : Elemento)
        Sacar( ES E: Elemento)
        primero()
 Fin
 Constructor Cola.Crear
  inicio
       // Crear Objeto L
   fin
Cola.Poner( E: Elemento)
Inicio
   l.inserta(l.primero(),E);
Fin

Cola.Sacar( ES E: Elemento)
Inicio
l.recupera(l.fin(),E)
l.suprime(l.fin)
Fin
```

20,99

L <99,20>

E=10

4.4.2 Implementacion basada en vectores V.1

```
Clase Cola
    Atributos
        V :     Arreglo(MAX)
       ini,fin: Entero
    Metodos
       Crear()
       Vacia()
       Poner(E : Elemento)
       Sacar( ES E: Elemento)
       primero()
Fin
Constructor Cola.Crear
  inicio
       fin=0
       ini=1
   fin
 Cola.Poner( E: Elemento)

 inicio
     si fin<MAX entoces
                fin = fin +1
                 v[fin]=E
 Fin
Cola.Sacar( ES E: Elemento)
Inicio
Si no Vacia() entices
    E= V[ini]
    ini=ini +1
Fin
```

10,20,33,44,7

V

10	20	33	44	7
1	2	3	4	5

INI 1

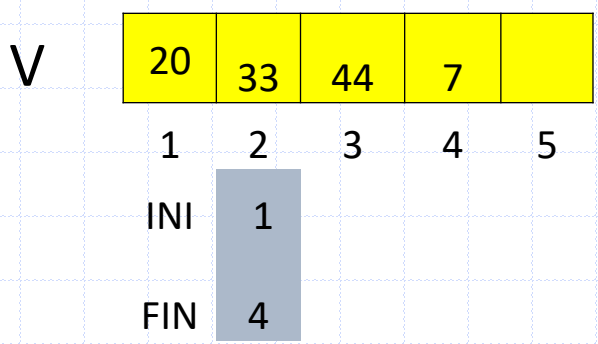
FIN 5

4.4.2 Implementacion basada en vectores V.2

```
Clase Cola
    Atributos
        V :     Arreglo(MAX)
       ini,fin: Entero
    Metodos
       Crear()
       Vacia()
       Poner(E : Elemento)
       Sacar( ES E: Elemento)
       primero()
Fin
Constructor Cola.Crear
  inicio
       fin=0
       ini=1
   fin
 Cola.Poner( E: Elemento)

 inicio
     si fin<MAX entoces
                fin = fin +1
                 v[fin]=E
 Fin
Cola.Sacar( ES E: Elemento)
Inicio
Si no Vacia() entices
    E= V[1]
    desplazar(1)
    fin = fin -1
Fin
```

20,33,44,7



4.4.2 Implementacion basada en vectores

(Vector de recorrido 1..MAX,1..Max,1..Max.. V.3)

```
Clase Cola
    Atributos
        V :      Arreglo(MAX)
       ini,fin: Entero
    Metodos
       Crear()
       Vacia()
       Poner(E : Elemento)
       Sacar( ES E: Elemento)
       primero()
 Fin
 Constructor Cola.Crear
  inicio
      fin=0    ini=1
   fin
 Cola.Poner( E: Elemento)
 inicio
   si  Siguiente(Siguiente(Fin))<>ini entoces
       fin = siguiente(fin)
       v[fin]=E
Fin
Cola.Sacar( ES E: Elemento)
Inicio.
    Si no Vacia() entoces
       E= V[ini]
       ini=siguiente(ini)
 Fin
```

44,12,88,99

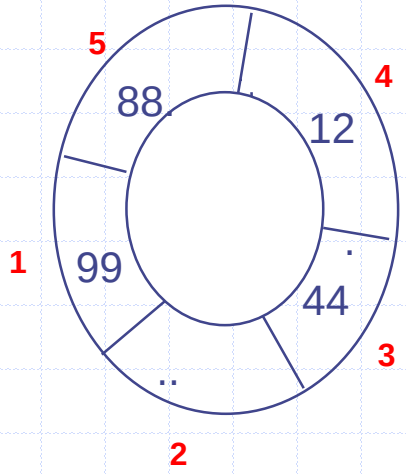
E=99

V

99		44	12	88
1	2	3	4	5

INI 3

FIN 1



4.4.2 Implementacion basada en vectores

(Vector de recorrido 1..MAX,1..Max,1..Max.. V.3)

La funcion Siguiente(Dir)

Esta funcion tiene por objetivo recibir una direccion y retornar la siguiente o la direccion que se encuentra a continuacion.

Implementacion

```
Siguiente ( dir )
Inicio
     si dir = MAX entonces retornar 1
                    caso contrario retornar dir+1
Fin
```

Si no se desea implementar la funcion siguiente se podria reemplazar esta por

```
dir = (dir modulo MAX) +1
```

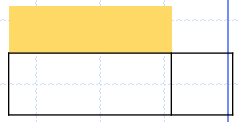
4.4.3 Implementación con apuntadores

Definición básica de la clase Cola con representación enlazada con simple enlace:

```
Tipo de dato
 Nodo
     elemento TipoElemento
     sig      Puntero a Nodo
 // fin definicion

 Direccion Puntero a espacio de memoria de tipo Nodo

Clase Cola
  Atributos
      ini Puntero de tipo Direccion
      fin Puntero de tipo Direccion
  Metodos
        ……..
```



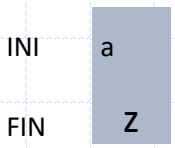
4.4.3 Implementación con apuntadores

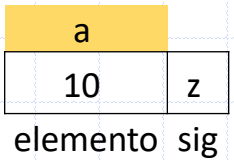
```
publico Cola.Crear()
    inicio
        ini = nulo ; fin = nulo
    fin
publico  Cola.poner( E : Elemento)
     inicio
        aux = // Pedir Espacio de memoria para Nodo
        si aux <> nulo entoces
                        ponerdato(aux,'->elemento',E);
                        Ponerdato(aux, '->sig',nulo)
                       si vacia() entoces
                                    ini=aux
                                    fin=aux
                                    caso contrario
                                     ponerdato(fin, '->sig',aux)
                                     fin = Aux
                caso contrario // Error
    fin
```

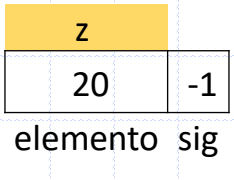
```
publico Cola.vacia()
  inicio
        retornar (ini = nulo)
  fin
```

Aux=z
E=20

10,20







4 Comparación de las Implementaciones

- La elección de una implementación u otra dependerá de los requerimientos de la aplicación en la que se use.
- Es posible obtener representaciones para colas que permiten operaciones eficientes con tiempos de ejecución constantes.
- Queda demostrado con la reprecentacion con vectores que las formas de implementacion deben ser totalmente agenas al usuario.

laboratorio

- Implementar vector recorriendo v.2
- Implementar con vector circular v.3
- Implementar con lista
- Implementar con simulación de memoria
- Implementar con punteros

laboratorio

- Implementar vector recorriendo v.2
- Implementar con vector circular v.3
- Implementar con lista
- Implementar con simulación de memoria
- Implementar con punteros