ESTRUCTURAS LINEALES (V 3.0)

Prof. José Fager – Paysandú agosto de 2017

fic

IPAD I

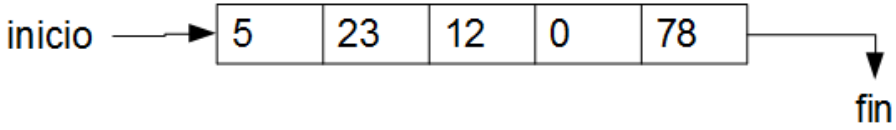
Listas

Definición

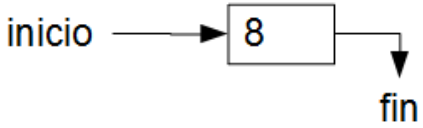
- Las listas son estructuras que organizan datos en forma encadenada.
- Existe un orden de ubicación de los datos dentro de la lista.
- Existe un primer elemento y un último bien determinados.
- También existe la lista vacía.

Ejemplo

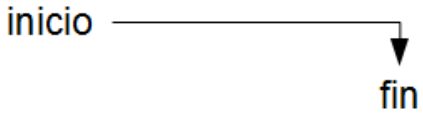
Lista con 5 elementos



Lista con 1 solo elemento



Lista vacía



TAD Lista

Definición

- El “Tipo Abstracto de Datos” (TAD) Lista, es un concepto matemático básico que define un tipo de dato.
- Está formado por datos (estructuras de datos) y las operaciones (procedimientos o funciones) que se realizan sobre esos datos.
- El TAD Lista nos permitirá implementar y manejar las diferentes estructuras de datos que veremos más adelante.

Operaciones del TAD Lista básico

//*** CONSTRUCTORAS ***//

//Retorna la lista vacía
Crear() ⟶ lista

//Inserta el dato "d" en la lista "l" al principio
Mete(l, d)

//Inserta el dato "d" en la lista "l" al final
Etem(l, d)

//Quita el dato al principio de la lista "l"
//Si "l" está vacía no se modifica
Quita(l)

//*** PREDICADO ***//

//Retorna true si la lista "l" es vacía sino retorna false
EsVacía(l)⟶ bool

// *** SELECTORAS ***//

//Retorna el dato al principio de la lista "l"
//Si "l" está vacía no retorna nada
Inicio(l) ⟶ dato

//Retorna la lista "l" sin su primer elemento
//Si "l" está vacía retorna la lista vacía
Resto(l) ⟶ lista

Pilas

Definición

- Las Pilas son estructuras del tipo LIFO (Last In First Out) y se implementan como listas.
- Su nombre proviene de una analogía con las pilas de platos para lavar en los restaurantes.
- El último elemento agregado a la Pila, es el que queda en primer lugar para ser sacado, es decir que el último en entrar es el primero en salir (LIFO).

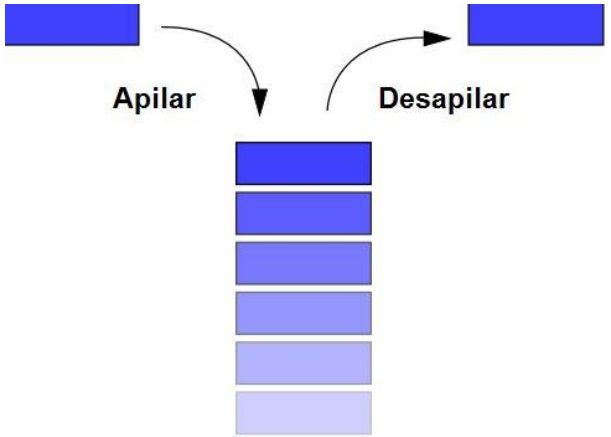


Imagen basada en: https://commons.wikimedia.org/wiki/File:Data_stack.svg

Semántica de operaciones en las Pilas

- **Crea:** Crea la pila vacía y la retorna.
- **Apilar:** Inserta un dato en la pila, **siempre** al principio de la pila.
- **Desapilar:** Quita un dato de la pila, que **siempre** es el que está al principio de la pila y retorna dicho dato. Si la pila está vacía no se modifica y no retorna nada.
- Para implementar: Crea, Apilar y Desapilar vamos a usar las operaciones del TAD Lista.

Operaciones de las Pilas

//Retorna la pila vacía

Crea() → pila

//Inserta el dato “d” al principio de la pila “p”

Apilar(p, d)

//Quita y retorna el dato al principio de la pila “p”

Desapilar(p) → dato

Implementación de operaciones en Pilas

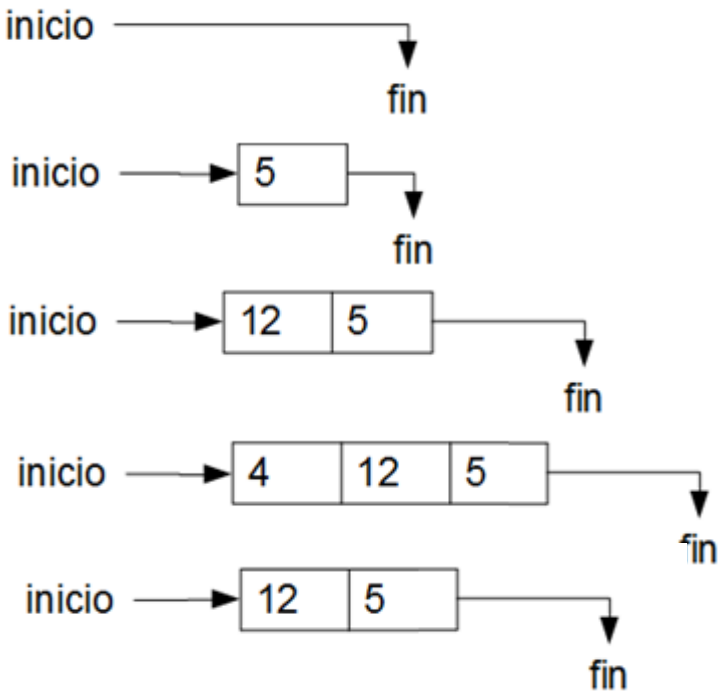
```
//Retorna la pila vacía
Crea() → pila
{
  //crea la pila vacía
  p:= Crear();
  //retorna la pila creada
  return p;
}

//Inserta el dato "d" al principio de la pila "p"
Apilar(p, d)
{
  //Inserta el dato "d" al principio de la pila
  Mete(p, d);
}
```

```
//Quita y retorna el dato al principio de la pila "p"
Desapilar(p) → dato
{
  //Toma el dato al inicio de la pila
  d:= Inicio(p);
  //Quita el dato al inicio de la pila
  Quita(p);
  //Retorna el dato que estaba al inicio de la pila
  return d;
}
```

Ejemplos

p:= Crea()	(crea la pila vacía)
Apilar (p, 5)	(inserta 5 en la pila)
Apilar (p, 12)	(inserta 12 en la pila)
Apilar (p, 4)	(inserta 4 en la pila)
d:= Desapilar (p)	(quita el tope y retorna el 4)



Cola

Definición

- Las Colas son estructuras del tipo FIFO (First In First Out) y se implementan como listas.
- Su nombre proviene de una analogía con las filas que se forman a la espera de ser atendidas.
- El primer elemento agregado a la Cola, es el que queda en primer lugar para ser sacado, es decir que el primero en entrar es el primero en salir (FIFO).

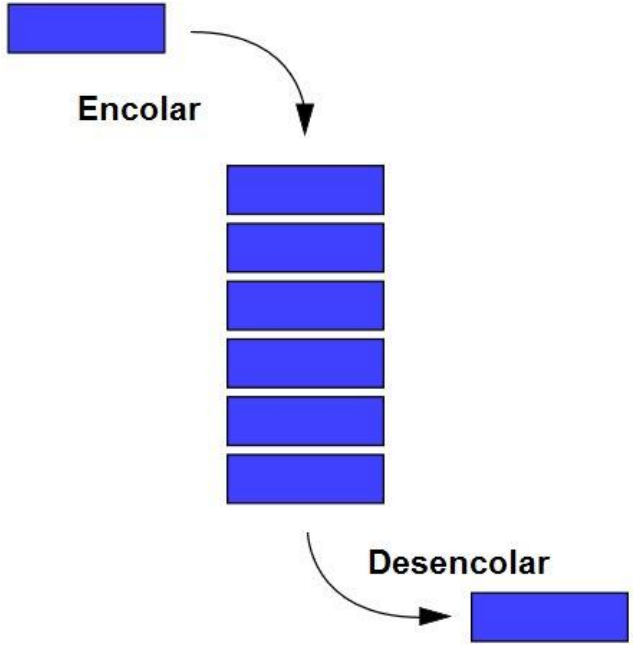


Imagen basada en: https://upload.wikimedia.org/wikipedia/commons/3/34/Fifo_queue.svg

Semántica de operaciones en las Colas

- **Crea:** Crea la cola vacía y la retorna.
- **Encolar:** Inserta un dato en la cola, **siempre** al final.
- **Desencolar:** Quita un dato de la cola, que **siempre** es el que está al principio (NO retorna dicho dato). Si la cola está vacía no se modifica.
- **Cabeza:** Retorna el dato al principio de la cola (pero NO lo quita). Si la cola está vacía no retorna nada.
- Para implementar: Crea, Encolar, Desencolar y Cabeza vamos a usar las operaciones del TAD Lista.

Operaciones de las Colas

//Retorna la cola vacía
Crea() → cola

//Inserta el dato "d" al final de la cola "c"
Encolar(c, d)

//Quita el dato al principio de la cola "c"
//Si la cola está vacía no se modifica
Desencolar(c)

//Retorna el dato al principio de la cola "c"
//Si la cola está vacía no retorna nada
Cabeza(c) → dato

Implementación de operaciones en Colas

```
//Retorna la cola vacía
Crea() → cola
{
  //crea la cola vacía
  c:= Crear();
  //retorna la cola creada
  return c;
}

//Inserta el dato "d" en la cola "c"
Encolar(c, d)
{
  //Inserta el dato "d" al final de la cola
  Etem(c, d);
}
```

```
//Quita el dato al principio de la cola "c"
//Si la cola está vacía no se modifica
Desencolar(c)
{
  //Quita el dato al principio de la cola
  Quita(c);
}

//Retorna el dato al principio de la cola "c"
//Si la cola está vacía no retorna nada
Cabeza(c) → dato
{
  //Toma el dato al inicio de la cola
  d:= Inicio(c);
  //Retorna el dato que estaba al inicio de la cola
  return d;
}
```

Ejemplos

c:= Crea() *(crea la cola vacía)*

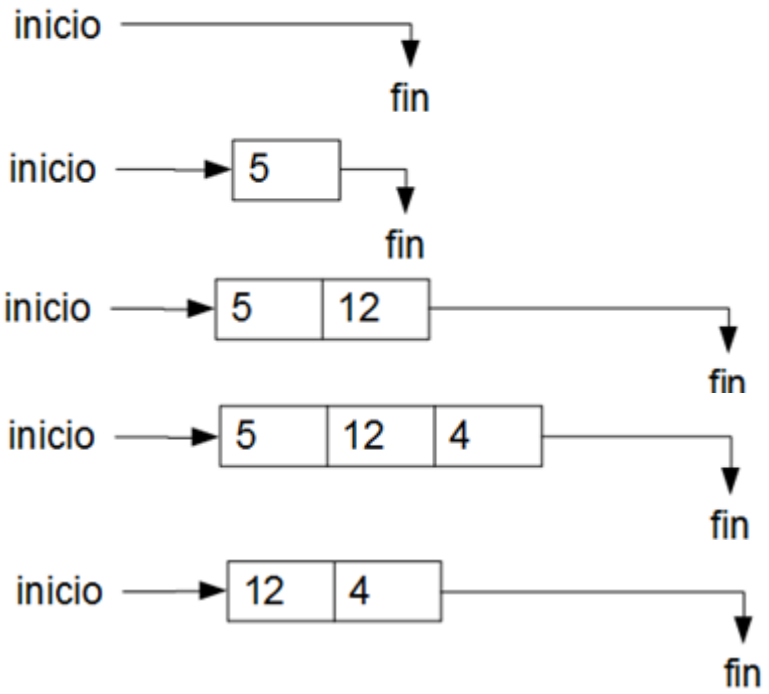
Encolar (c, 5) *(inserta 5 en la cola)*

Encolar (c, 12) *(inserta 12 en la cola)*

Encolar (c, 4) *(inserta 4 en la cola)*

Desencolar (c) *(quita 5 de la cola)*

d:= Cabeza(c) (retorna 12)



Colas de prioridad

Definición

- Son similares a las colas con la diferencia que los elementos no se insertan en función de su orden de llegada sino que considerando una prioridad.
- Por ejemplo un concepto de prioridad podría ser insertar el elemento de manera que siempre se mantenga una relación de orden.

Ejemplo

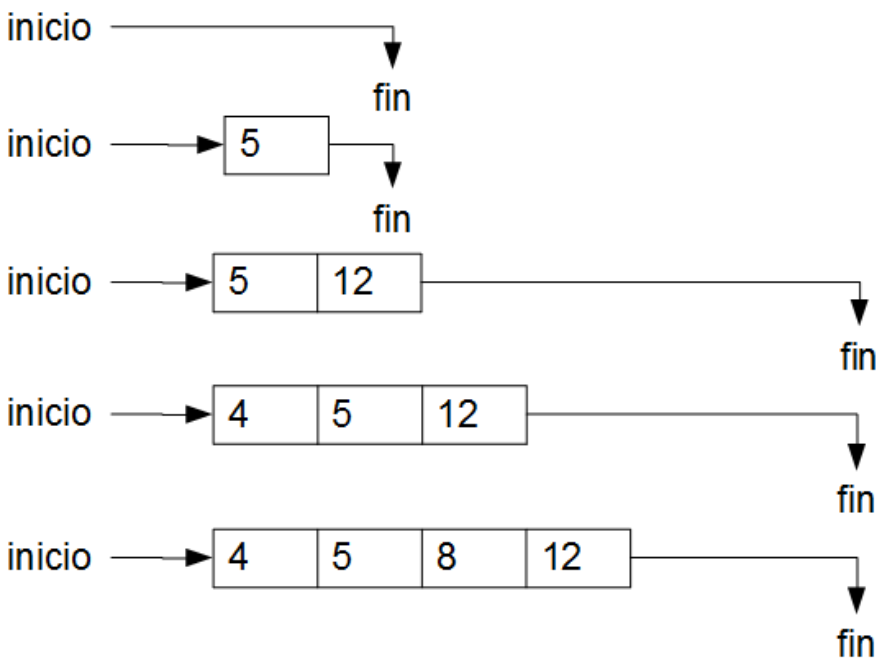
c:= Crea()

Encolar(c,5)

Encolar(c,12)

Encolar(c,4)

Encolar(c,8)



La prioridad en este ejemplo es: “ordenar de menor a mayor”