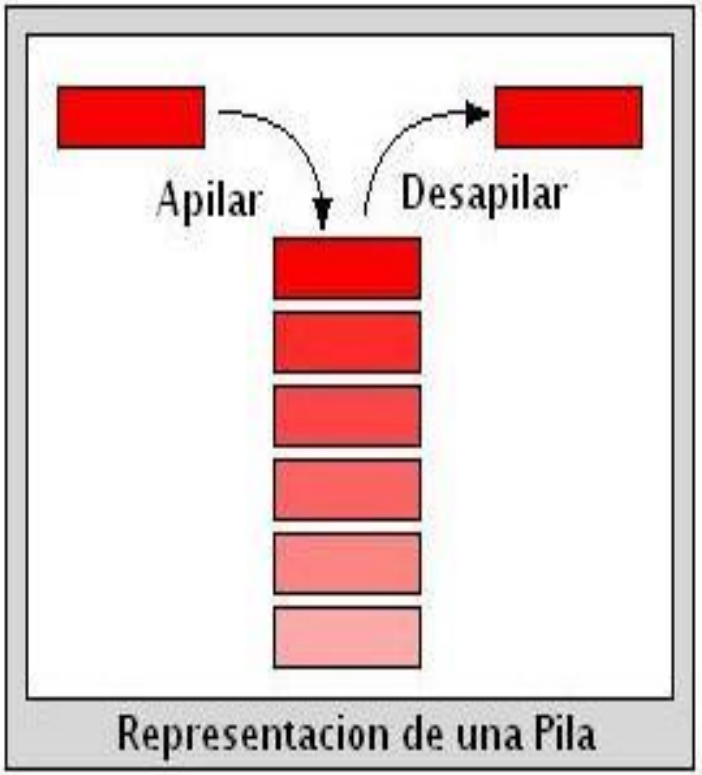
UNIDAD 2: ESTRUCTURAS DE DATOS DINÁMICAS

Pilas y colas

PILAS

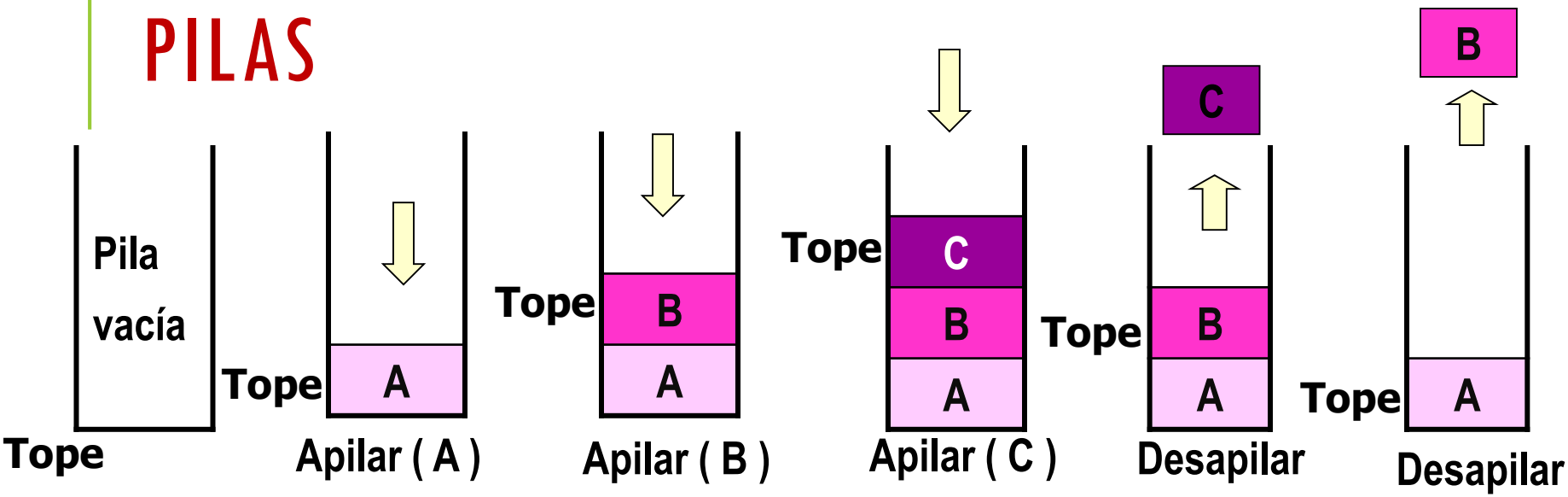
Una pila (o stack) es una estructura de datos del tipo **LIFO (Last In First Out)**, es decir, lo último en entrar es lo primero en salir.

Es una lista de elementos caracterizada porque las operaciones de inserción (apilar) y eliminación (desapilar) se realizan solamente por un único extremo de la estructura, denomino habitualmente tope (o cima).



Representacion de una Pila

PILAS



Algunos ejemplos de estructuras tipo LIFO son:

- Montón de platos
- Papas fritas Pringles
- Pila de libros

APLICACIONES DE PILAS

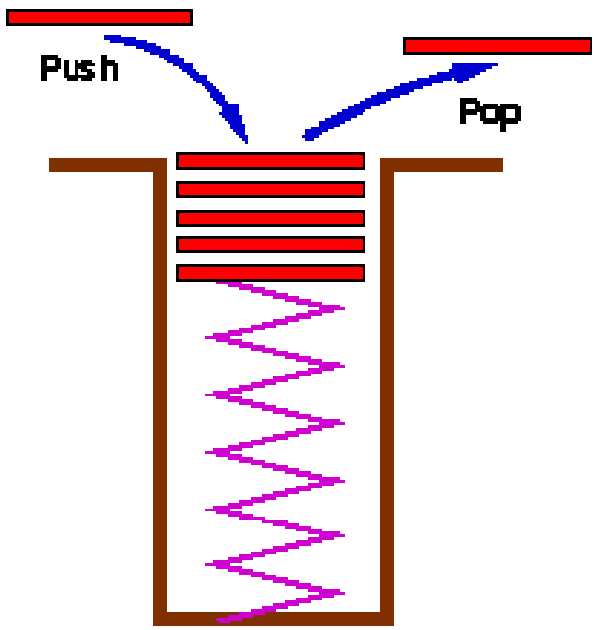
Algunos ejemplos del uso de pilas son:

- Los navegadores en Internet almacenan en una pila las direcciones de los sitios más recientemente visitados.
- Los editores de texto proporcionan normalmente un botón deshacer que cancela las operaciones de edición recientes y restablece el estado anterior del documento.
- En compiladores
  - Analizador sintáctico de símbolos equilibrados
  - Evaluación de expresiones postfijas

PILAS

Asociadas con la estructura pila existen una serie de operaciones necesarias para su manipulación, éstas son:

- Crear la pila
- Comprobar si la pila está vacía
- Apilar (Push): añadir elementos en el tope
- Desapilar (Pop): eliminar elementos del tope
- Espiar (peek): obtener el elemento en el tope **sin** eliminarlo de la pila



REPRESENTACIÓN DE PILAS

Los lenguajes de programación procedurales no suelen disponer de un tipo de datos **pila**. Por lo tanto, en general, es necesario representar la estructura pila a partir de otros tipos de datos existentes en el lenguaje

- Los lenguajes orientado a objetos generalmente si disponen de la clase stack (pila) como parte de su biblioteca.

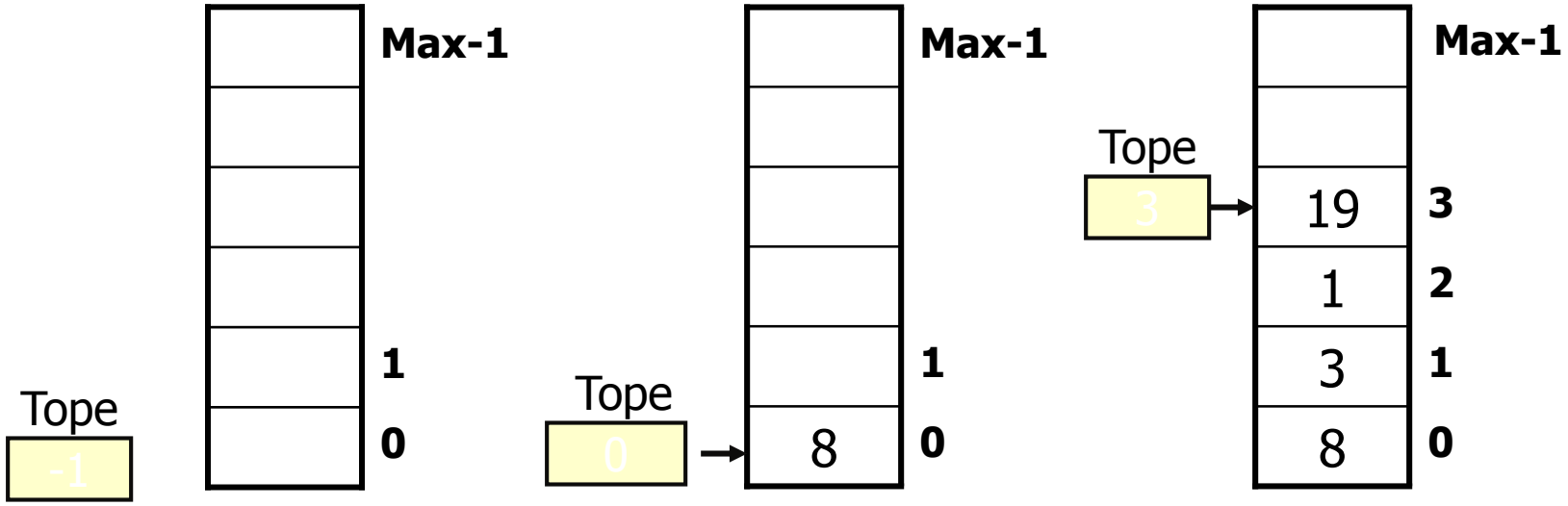
La forma más simple de representar una pila es mediante un arreglo unidimensional.

También es posible implementar una pila usando una lista enlazada.

REPRESENTACIÓN ESTÁTICA DE UNA PILA (USANDO ARREGLOS)

Para el manejo de la pila se requiere de los siguientes elementos:

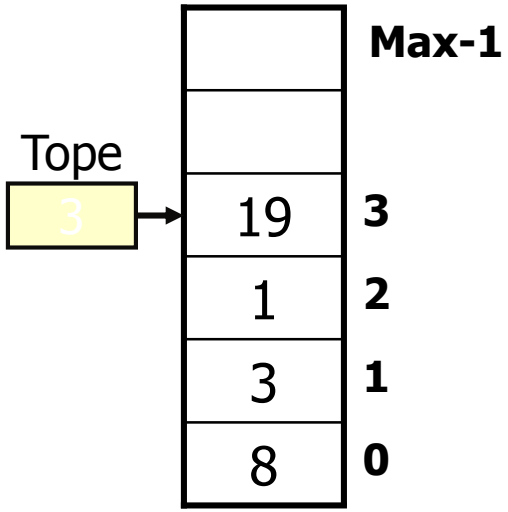
- Un vector (de algún tipo) para almacenar los datos de la pila
- Un entero (variable tope)



Tope = -1 (pila vacía)

CLASE PILA

¿Qué requerimos para modelar una pila de enteros de tamaño n?



Pila
- []datos: int - max: int - tope: int
+ Pila( ) + Pila(tam) + estaVacia(): boolean - isEmpty + estaLlena(): boolean - isFull + push(int): boolean + pop(): int + peek(): int //obtener elemento sin sacarlo

ACTIVIDAD COLABORATIVA

En binas, implementen los métodos de la clase PILA usando un arreglo de enteros.

Probar esta clase en un programa sencillo que realice los siguiente:

- Muestre un menú con las siguientes opciones
  - Apilar un elemento (push)
  - Desapilar un elemento (pop) (mostrarlo en pantalla)
  - Mostrar un elemento en el tope (sin sacarlo de la pila

ACTIVIDAD COLABORATIVA

En binas, resuelvan el siguiente problema: Supongamos que *TEST* es alguna función Booleana que recibe como parámetro cualquier dato entero y devuelve un valor **TRUE** o **FALSE**. Considera el siguiente segmento de código.

¿Cuáles de las siguientes opciones NO es una posible salida del código anterior?.

a)1 2 3
b)1 3 2
c)3 1 2
d)2 3 1

```
for ( i=1; i<=3; i++) {
   if (!TEST(i))
       p.push(i);
  else
       System.out.println( i +" ");
}
while (!p.esVacia()){
   x = p.pop();
   System.out.println( x+" ");
}
```

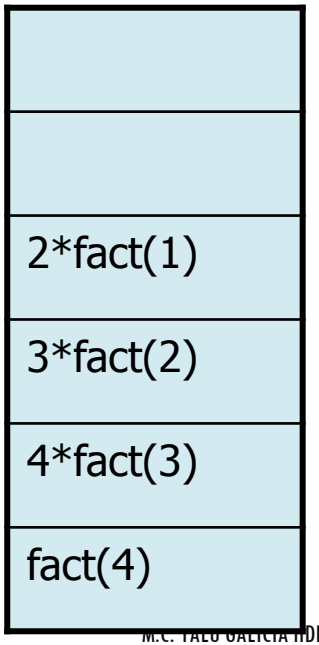
APLICACIONES DE PILAS

APLICACIONES DE PILAS

Funciones recursivas

- Las pilas pueden ser usadas para implementar la recursión en programas
- Una función o procedimiento recursivo es aquel que se llama a sí mismo
- Ejemplos:
  - Factorial, Fibonacci, Torres de hanoi

```
int Fact(n) {
  if(n == 1 || n == 0)
          return 1;
  return n*fact(n-1);
}
```



ANALIZADOR DE SÍMBOLOS

Problema:

- Dado una secuencia de caracteres (arreglo de caracteres), comprueba si los caracteres '{', '[', '<' y '(' se encuentran equilibrados (emparejados) o no.
  - Ejemplo: ((][<>}})) - No equilibrado
  - Ejemplo: (([{}<>])) - equilibrado
- En otras palabras, toda llave de apertura '{', debe tener asociada una llave de cierre '}', todo '[' un ']' y si existe un '(' debe existir el ')'.
- Contar simplemente el número de apariciones de cada símbolo no es suficiente. Por ejemplo la secuencia [()] es correcta, pero [(]) no lo es.

ANALIZADOR DE SÍMBOLOS

En este problema se puede utilizar una pila porque sabemos que cuando encontremos un símbolo de terminación, por ejemplo '}', este debe emparejarse con el símbolo '{' anterior mas reciente aún no emparejado.

Por lo tanto, colocando los símbolos de apertura en una pila podemos fácilmente comprobar si la aparición de un símbolo de terminación es o no oportuna.

En concreto, tenemos el siguiente algoritmo para comprobar símbolos equilibrados:

ALGORITMO ANALIZADOR DE SÍMBOLOS

Crear una pila vacía (instancia de la clase Pila)

Leer una cadena de símbolos

Para cada caracter (o símbolo) en la cadena hacer

- Si el símbolo es de apertura, apilarlo (push)
- Si el símbolo es de cierre y la pila NO está vacía, desapilar (pop) el símbolo y comparar ambos símbolos.
  - Si el símbolo desapilado no es el correspondiente símbolo de apertura, entonces no están emparejados y enviar mensaje "símbolos no equilibrados"
  - En caso contrario, si la pila está vacía, enviar el mensaje símbolos no equilibrados, sino continuar

Al finalizar la cadena, si la pila no está vacía, enviar el mensaje "simbolos no equilibrados". De lo contrario escribir "Simbolos equilibrados" y terminar

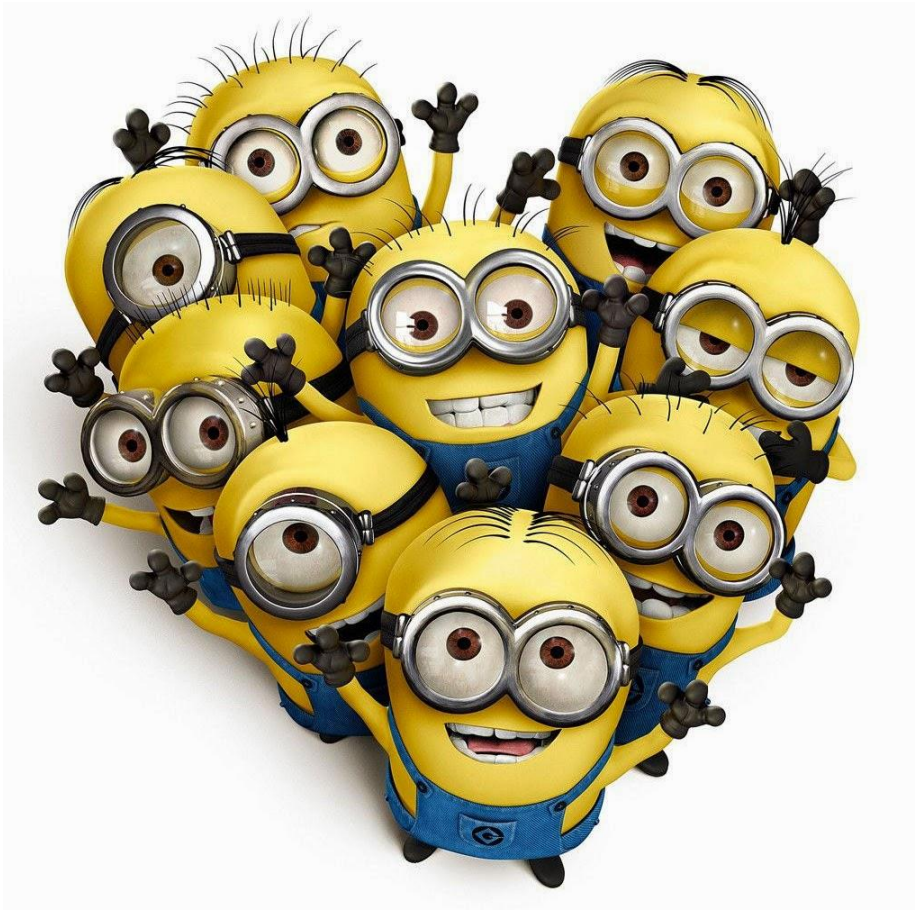
ACTIVIDAD COLABORATIVA

En binas trabaja con el problema del analizador de símbolos.

A continuación se muestra el diagrama que se sugiere para la clase Analizador

Analizador
- miPila: Pila
+Analizador() +analizarSimbolos(String cad): boolean - esSimApertura(char s): boolean - esSimTerminacion(char s): boolean - estanEmparejados(char sa,char st): boolean

¿QUE HEMOS APRENDIDO?



APLICACIONES DE PILAS

Evaluar Expresiones aritméticas

- Una expresión aritmética contiene constantes, variables y operaciones con distintos niveles de precedencia.
- Operaciones:
  - */ Multiplicación, división
  - +- Suma, resta

EJEMPLO

¿A qué es igual la siguiente expresión?

$$4/2 - 3 + 7 *3 - 6*2$$

- A) 24
- B) 8

NOTACIONES

Notación infija

Los operadores aparecen en medio de los operandos.

A + B, A – 1, E/F, A * C – 2, A+B*3

Notación prefija

El operador aparecen antes de los operandos.

+AB, -A1, /EF, *AC– 2, +AB-3

Notación posfija

Los operadores aparecen al final de los operandos.

AB+, A1-, EF/, AC*2-, AB-3-

CALCULADORA SENCILLA

En lugar de las dificultades que involucra evaluar las expresiones aritméticas en su notación **infija**, estás se convierten a notación **postfija**

Entre las ventajas de esto tenemos: no hace falta paréntesis, no hace falta definir prioridad de operadores y la evaluación final de la expresión se realiza fácilmente con un simple recorrido de izquierda a derecha de la expresión. Lo que proporciona un mecanismo directo de evaluación, en donde el operador se sitúa detrás de los operandos sobre los que actúa.

Ejemplo: 93/5*2- -> (9/3*5-2 infija)

PASOS PARA EVALUAR UNA EXPRESIÓN

1. Convertir a posfijo
   - Convertir la expresión de notación infija a posfija
2. Evaluar la expresión posfija
   - Usar una pila para mantener los resultados intermedios cuando se evalúa la expresión en notación posfija

ALGORITMO INFIJO-POSTFIJO

```
Inicio
   P <-- Crear() // crea pila
   Mientras (no es el fin de la expresion ) hacer
       x ← sig_elemento(expresion)
       si (x es operando) entonces
           enviar  a salida x
       sino
           //analizamos prioridades de operadores
           Mientras (!P. vacia()) && (prioridad(x)<=prioridad(P.mirar()) hacer
                y ← P.desapilar()
                envía a salida y
            fin_mientras
            P.apilar(x)
       fin_si
   fin_mientras
```

ALGORITMO INFIJO-POSTFIJO

```
    Mientras (! P.vacia( )) hacer
          y ← P.desapilar()
          salida(y)
       fin_mientras
Fin
```

ACTIVIDAD COLABORATIVA

En binas, convertir las siguientes expresiones aritméticas de notación infija a postfija usando el algoritmo mencionado

- 2 -1 + 4 + 7
- 5 * 2 + 9 / 3 – 4 * 2 * 5 - 1

EVALUACIÓN DE EXPRESIONES EN NOTACIÓN POSFIJA

Para realizar el cálculo de las expresiones en notación postfija, hay que tener en cuenta que al leerlas de izquierda a derecha, lo primero que se lee son los operandos y después el operador. Por ello, es necesario almacenar la información leída hasta que se determine que operador hace uso de ella.

Además, los operadores actúan sobre los últimos operandos leídos. De manera que, conviene recuperar la información en sentido inverso a como se almacena.

Por esa razón, parece natural emplear una pila como estructura de almacenamiento de información

ALGORITMO PARA EVALUAR

```
p: pila
Inicio
        leer expresion posfija
        mientras (hay elementos en la expresion) hacer
            si elemento = operando entonces
                    p.apilar(elemento)
         sino //es un operador
             der = p.desapilar( ) // 1er. operando
              izq = p.desapilar() // 2do. operando
              aplicar operador sobre los operandos desapilados
              p.apilar(resultado)
         fin_si
    fin_mientras
     res = p.desapilar() //desapilar resultado
     escribir res
Fin
```

ACTIVIDAD COLABORATIVA

En binas, evaluar las expresiones en notación a postfija obtenidas en la actividad anterior, aplicando el algoritmo correspondiente

CALCULADORA SENCILLA

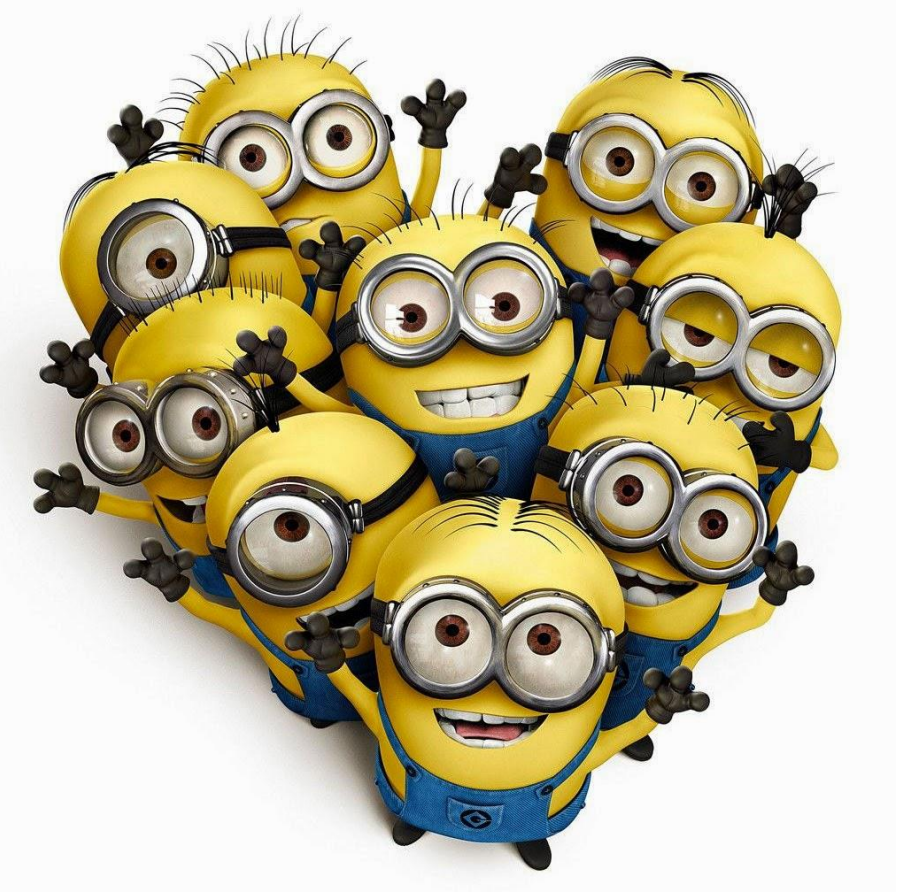
Nota que la calculadora, requiere de dos pilas diferentes. Es decir, una pila de operadores (+,/,*,-) para la conversión a notación postfija y una pila de operandos (números) para la evaluación de la expresión.

TIP: Para convertir un carácter a entero, simplemente al carácter réstale el ascii del 0.

- Ejemplo: si c = '5' para convertir a dígito realiza lo siguiente:
- int d = c- '0' => d = 5

Recuerda que el arreglo postfijo es de caracteres y al evaluar deben ser números.

¿QUE HEMOS APRENDIDO?



COLAS (QUEUES)

Estructura de Datos

COLAS

Una cola (o queue) es una lista de elementos en la cual los elementos se insertan por un extremo (el final) y se eliminan por el otro (el inicio).

Un cola es una estructura de datos del tipo FIFO (First In First Out). Esto es, el primero en entrar es el primero en salir.

Los elementos salen en el mismo orden en que entraron.

APLICACIÓN DE LAS COLAS

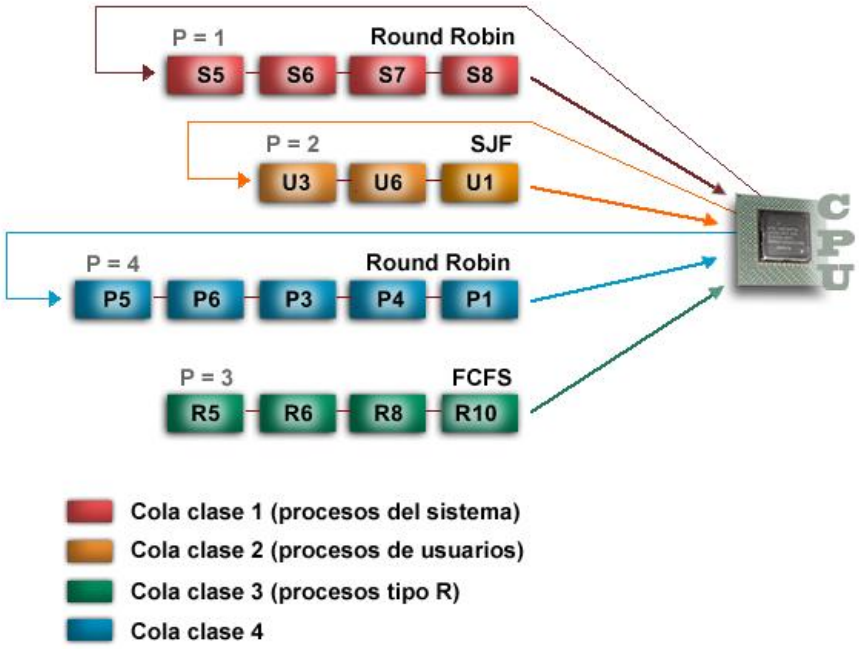
Las colas, al igual que las pilas, resultan de aplicación habitual en muchos problemas informáticos.

Quizás la aplicación más común de las colas es la organización de tareas de una computadora.

- En general, las tareas enviadas al procesador son "encoladas" por este, para ir procesando secuencialmente todos los trabajos en el mismo orden en que se reciben.

APLICACIÓN DE LAS COLAS

Cuando el procesador recibe el encargo de realizar una tarea, ésta es almacenada al final de la cola de trabajos. En el momento que la tarea que estaba realizando el procesador acaba, éste selecciona la tarea situada al principio de la cola para ser ejecutada a continuación.



APLICACIÓN DE LAS COLAS

Del mismo modo, es necesaria una cola, por ejemplo, a la hora de gestionar eficientemente los trabajos que deben ser enviados a una impresora (o a casi cualquier dispositivo conectado a una computadora). De esta manera, la computadora controla el envió de trabajos al dispositivo, no enviando un trabajo hasta que la impresora no termine con el anterior

COLAS

Asociadas con la estructura Cola existen una serie de operaciones necesarias para su manipulación, éstas son:

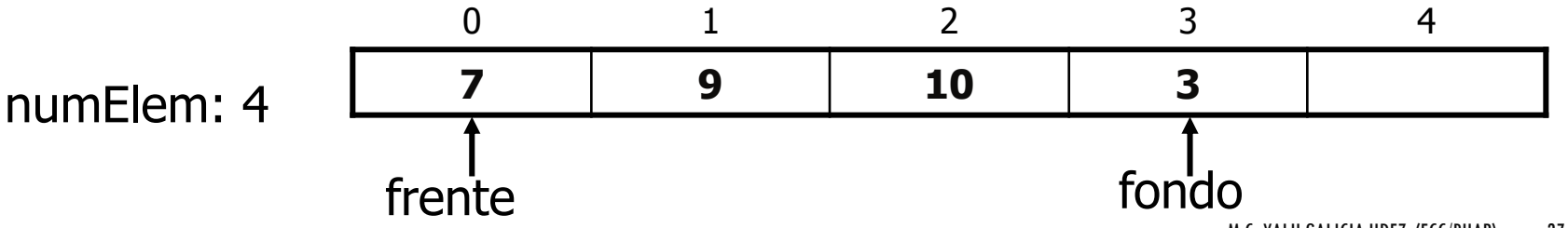
- Crear una cola vacía
- Determinar si la cola está vacía, en cuyo caso no es posible eliminar elementos
- Insertar elementos al final de la cola (encolar - queue)
- Eliminar elementos al inicio de la cola (desencolar - dequeue)
- Acceder al elemento inicial de la cola sin desencolarlo (peek)

REPRESENTACIÓN DE COLAS

La representación de una cola lineal finita en forma de vector es una tarea algo más compleja que la realizada para el caso de las pilas.

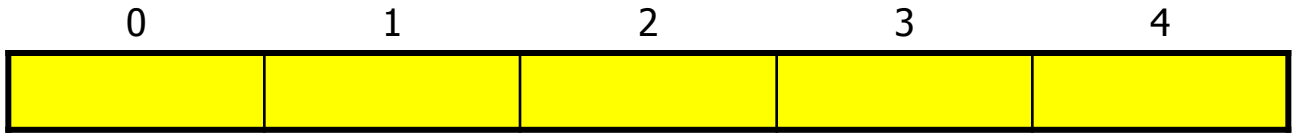
Además de un arreglo unidimensional, son necesarias un par de variables que indiquen dónde está el inicio de la cola y dónde el final.

- frente: es un índice que indica la posición del primer elemento de la cola, es decir, la posición del elemento a retornar al invocar el método desencolar()
- Fondo: es el índice de la posición del último elemento de la cola. Si se invoca encolar(), el elemento debe ser insertado en el casillero siguiente al que indica el fondo.
- numElem: indica cuantos elementos posee realmente la cola.

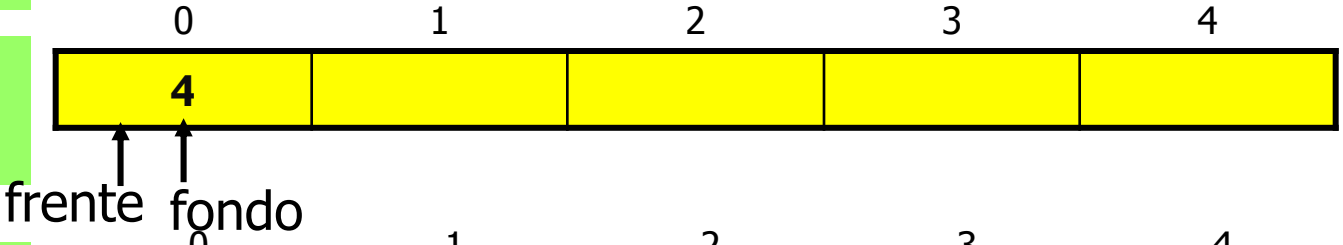


EJEMPLO

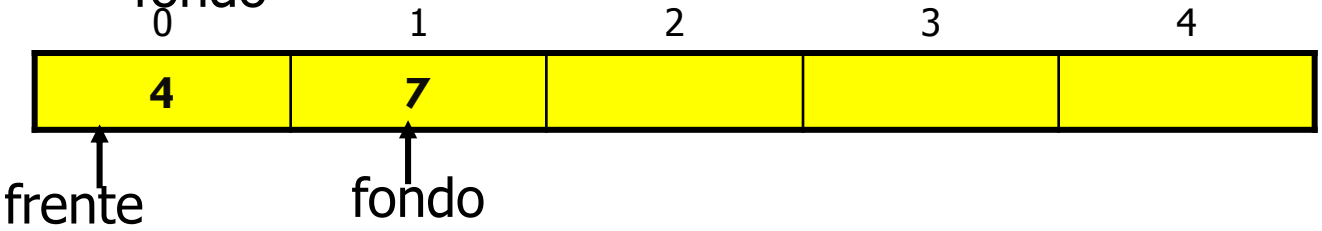
numElem: 0
Cola Vacía
Frente = 0
Fondo = 0



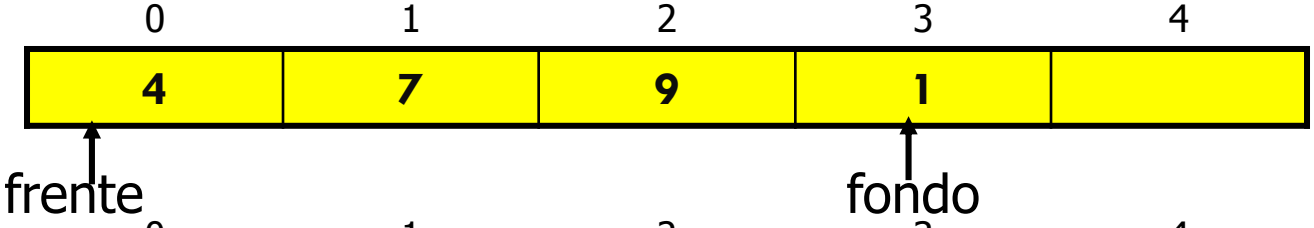
numElem: 1
Encolar: 4



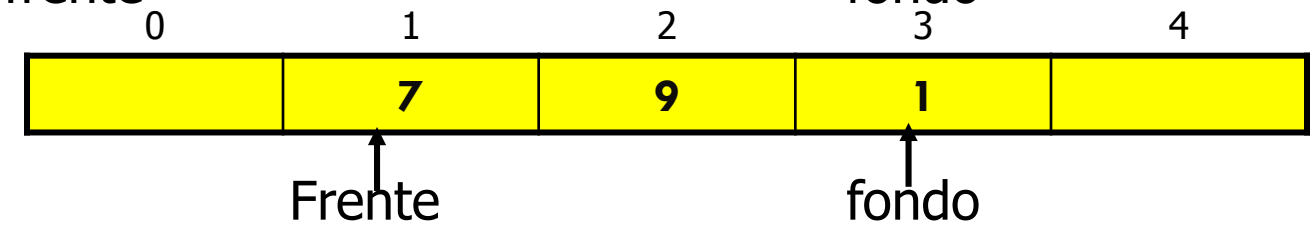
numElem: 2
Encolar: 7



numElem: 3,4
Encolar: 9, 1



numElem: 3
Desencolar: 4



CLASE COLA

¿Qué requerimos para modelar una cola de enteros de tamaño n?

Cola
- valores[]: int - max: int //numero de elementos máximo - frente, fondo: int
+Cola(tam: int) +Cola( ) + estaVacia(): boolean + estaLlena(): boolean + encolar(dato: int): void + desencolar(): int + espiar():int

ACTIVIDAD COLABORATIVA

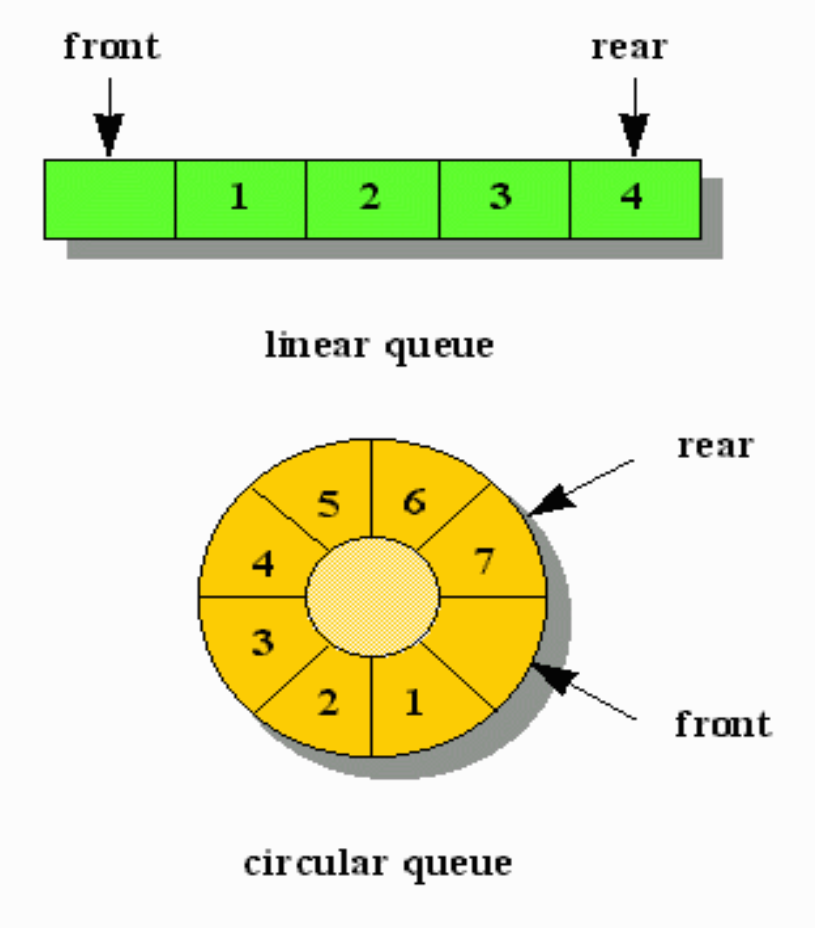
En bina, implementen los métodos de la clase Cola usando un arreglo de enteros.

Probar esta clase en un programa sencillo que realice los siguiente:

- Muestre un menú con las siguientes opciones
  - Encolar un elemento
  - Desencolar un elemento (mostrarlo en pantalla)
  - Espiar el elemento (sin sacarlo de la cola)

COLA CIRCULAR

Una de las formas más eficiente de implementar las operaciones relacionadas con colas, usando un arreglo estático, es representando la Cola[0..n-1] como si fuese circular, es decir, cuando se dé la condición de cola llena se podrá continuar por el principio de la misma si esas posiciones no están ocupadas.



COLA CIRCULAR

Para implementar una cola circular debemos considerar como actualizar las variables FONDO (rear) y FRENTE (front)

Consideremos los siguientes casos

Si la cola esta vacía

- Frente = 0 y Fondo = 0

Caso contrario

- Al encolar
  - fondo = (fondo+1)%tam
- Al desencolar
  - frente = (frente+1)%tam;

ACTIVIDAD COLABORATIVA

Modifiquen su clase para implementar ahora una Cola Circular usando un arreglo de enteros.

ACTIVIDAD COLABORATIVA

En binas, resuelvan el siguiente problema:

- Hacer un programa que reemplace un dato ya existente en una cola circular por una dato nuevo.
- No deben perderse los datos previamente existentes, esto es, si el primer valor aparece en algún lugar de la Cola, este debe ser reemplazado por el nuevo en el mismo lugar.

¿QUE APRENDIMOS?