Listas, Colas, y Pilas

Curso de Estructuras de Datos y Algoritmos I

Prof. Luis E. Garreta U.

lgarreta@uao.edu.co

Universidad Autonoma de Occidente – Cali
Depto. Operaciones y Sistemas
Facultad de Ingeniería

8 de mayo de 2018

Objetivo

- Desarrollar programas que incorporen las principales estructuras de datos lineales: Pila, Cola y Lista enlazada.
- Conocer, analizar y aplicar estructuras de datos dinámicas usando las bibliotecas de JAVA para la solución de problemas informáticos.

Introducción

- En muchas ocasiones se necesitan estructuras que puedan cambiar de tamaño durante la ejecución del programa.
- Se pueden usar arreglos dinámicos, pero una vez creados, su tamaño también será fijo, y para hacer que crezcan o disminuyan de tamaño, deberán reconstruirse desde el principio.
- Las estructuras dinámicas permiten:
  - ✓ Crear estructuras de datos que se adapten a las necesidades reales a las que suelen enfrentarse los programas:
  - ✓ Crear estructuras de datos muy flexibles, ya sea en cuanto al orden, la estructura interna o las relaciones entre los elementos que las componen.

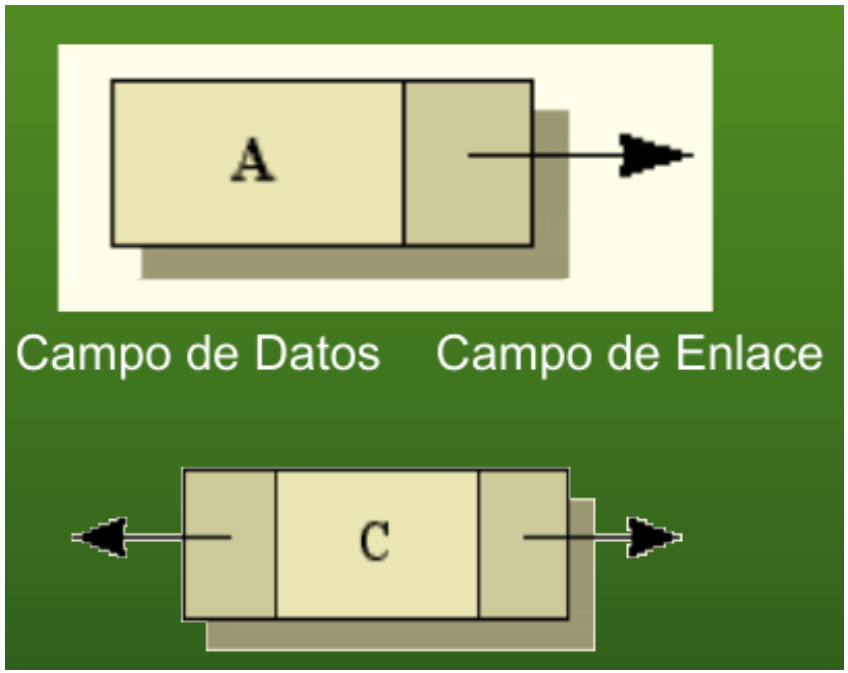
Tipos de Estructuras de Datos Lineales

- Lista – List
- Pila – Stack
- Cola – Queue

Nodos y Listas

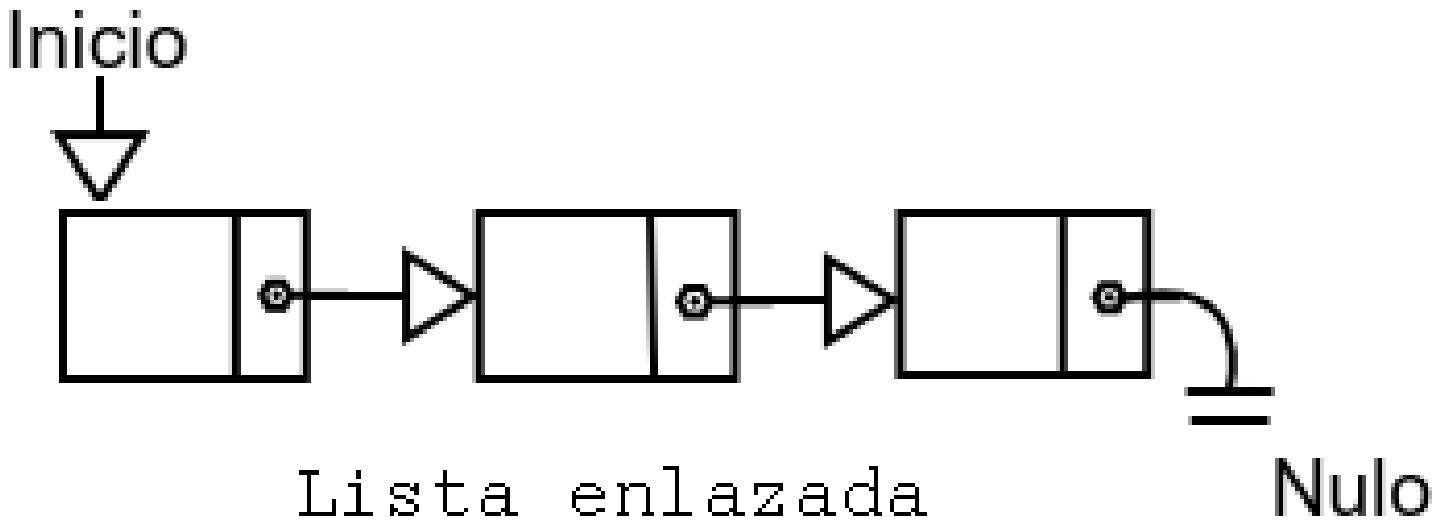
Los Nodos conforman a una lista

- Un nodo es una estructura que agrupa un conjunto de datos o campos arbitrarios, donde por lo menos un campo es de referencia o enlace a un nodo del mismo tipo (autorreferenciado).

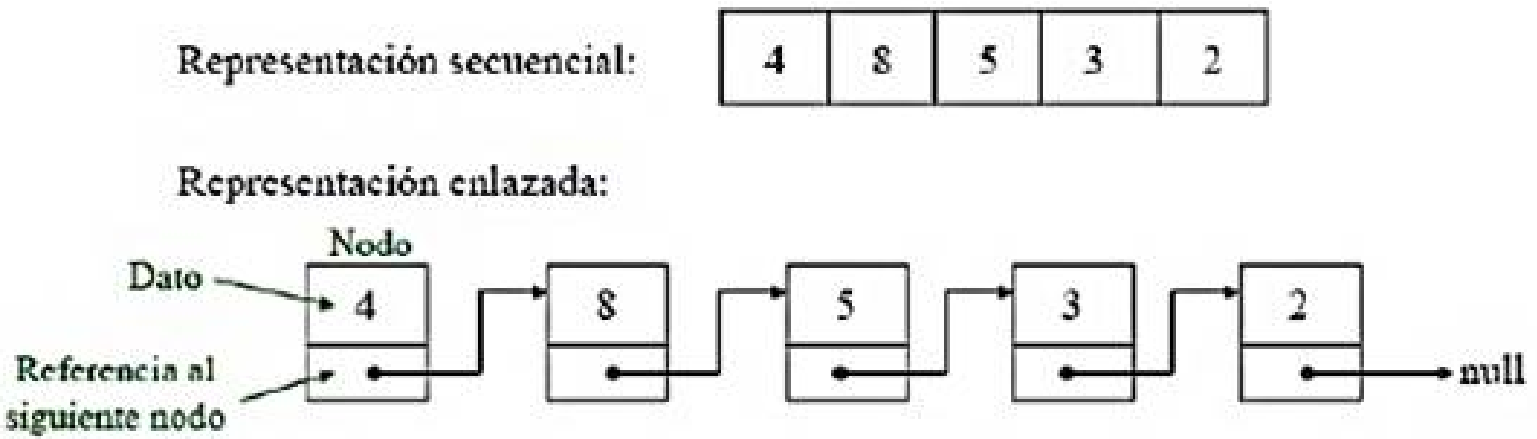


¿Qué es una Lista?

- Es una secuencia de nodos autorreferenciados con una o dos referencias al nodo anterior y/o siguiente.
- Una lista enlazada **(linked list)** es una de las estructuras de datos fundamentales y puede ser usada para implementar otras estructuras de datos, como pilas y colas.

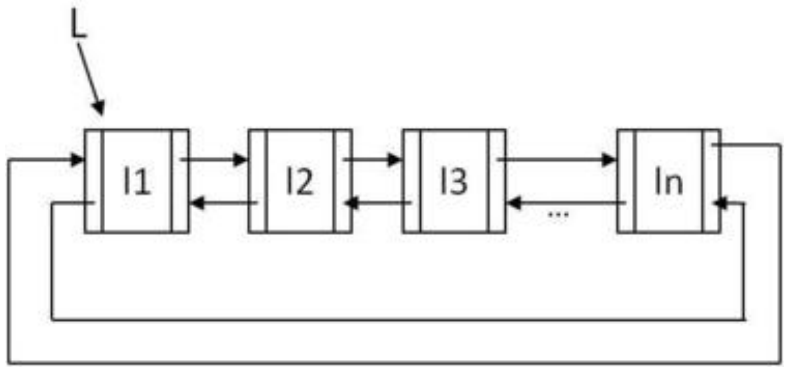


Otras Representaciónes de Listas



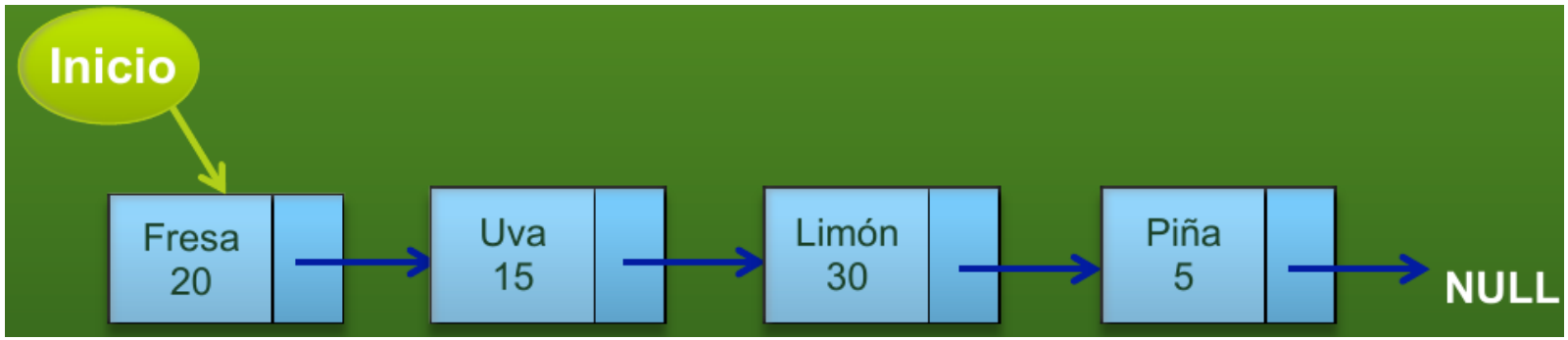
Tipos de Listas

- Lista simplemente enlazada
- Lista doblemente enlazada
- Lista circular simple
- Lista circular doble



Lista simplemente enlazada

- Cada nodo de la estructura tiene un único campo de enlace que apunta al siguiente nodo en la lista.
- El ultimo nodo en la lista apunta NULL (vacio).



Operaciones de las Listas

- Creación:
- Adición de Elementos
  - ✓ Generalmente al final
- Inserción de Elementos:
  - ✓ Al inicio de la lista
  - ✓ En medio entre dos nodos
  - ✓ Al final de la lista
- Borrado de Elementos;
  - ✓ Primer Nodo
  - ✓ Cualquier otro nodo
- Búsqueda de Elementos:
  - ✓ Desde el inicio
- Otras:
  - ✓ Ordenamiento

TADs: Tipos Abstractos de Datos

- Un tipo abstracto de datos (TAD) es un tipo definido por el usuario que:
  - ✓ Tiene un conjunto de valores y un conjunto de operaciones.
  - ✓ Cumple con los principios de abstracción, ocultación de la información y se puede manejar sin conocer la representación interna.
- TAD Lista

Lista
crear ()
adicionar (elemento)
insertar (elemento, posicion)
eliminar (posicion)
longitud () : posicion
obtener (posicion) : elemento
destruir ()

Operaciones Derivadas: función buscar elemento

función buscar

- Ingresa un elemento a buscar
- Retorna la posición, si lo encuentra, de lo contrario retorna -1

```
def buscar (elem:Elemento): posicion {
  n = longitud ()
  for (i=1; i <= n; i++) {
    elementoActual = obtener (i)
    if (elementoActual == elemento)
      return i
  }
  return -1
}
```

Operaciones Derivadas: concatenar lista

- funcion concatener lista
  - ✓ Ingresa otra lista
  - ✓ Adiciona los elementos de la otra lista a la lista actual

```
def concatenar (otraLista: Lista) {
  for (i=1; i < otraLista.longitud (); i++) {
    elem = otraLista.obtener (i)
    adicionar (elem)
  }
}
```

Operaciones Derivadas: obtener ultimo / eliminar último

- Función obtener el último elemento
  - ✓ Retorna el último elemento de la lista

```
def ultimo () {
  n = longitud ()
  elem = obtener (n)
  return elem
}
```

- Función eliminar el último elemento
  - ✓ Elimina el último elemento de la lista

```
def eliminarUltimo () {
  n = longitud ()
  eliminar (n)
}
```