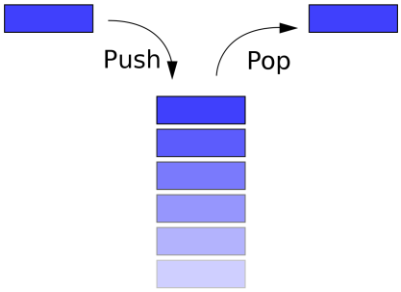
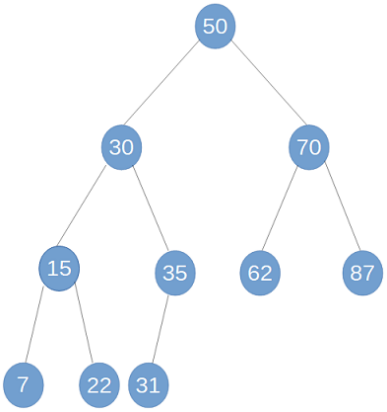
| | | |
|---|---|---|
| fidélitas UNIVERSIDAD | **Ingeniería en Sistemas de Computación** | **Curso:** Estructuras de Datos |
| | | **Preparado por:** Prof. MGT. Esteban Marín Ch. |
| | | **Código curso: SC-404** |

# Guía Rápida de Estudio de Estructuras de Datos

## Prof. MGT. Esteban Marín Chinchilla



Algoritmo para llenar una pila

```
 public void apilar(){
    1. Dato d=new Dato();
    2. d.setLetra(JOptionPane.showInputDialog(null,"Digite una
       letra:").charAt(0));
    3. Nodo nuevo=new Nodo();
    4. nuevo.setElemento(d);
    5. if(esVacia()){
    6.     cima=nuevo;
    7. }else{
    8.     nuevo.setSiguiente(cima);
    9.     cima=nuevo;
    10. }
}
```

1. Se crea el objeto.
2. Se llena el objeto.
3. Se crea el nodo nuevo.
4. Se almacena el objeto en el nodo.
5. Preguntar si está vacía.
6. Si está vacía, colocamos la cima en el nuevo nodo.
7. Si no
8. Enlazamos el nuevo elemento con la cima y
9. Finalmente, colocamos cima en el nuevo elemento.

Algoritmo para llenar una cola

```
public void encolar(){
    1. Dato d = new Dato();
    2. d.setLetra(JOptionPane.showInputDialog(null,
                  "Digite una letra:").charAt(0));
    3. NodoCola nuevo = new NodoCola();
    4. nuevo.setElementoCola(d);
    5. if (esVaciaCola()) {
    6.     inicio = nuevo;
    7.     fin = nuevo;
    8. } else {
    9.     fin.setSiguiente(nuevo);
    10.    fin = nuevo;
    11. }
}
```

1. Se crea el objeto.
2. Se llena el objeto.
3. Se crea el nodo nuevo.
4. Se almacena el objeto en el nodo.
5. Preguntar si está vacía.
6. Si está vacía, colocamos el inicio en el nuevo nodo.
7. Colocamos fin en el nuevo nodo.
8. Si no
9. Enlazamos fin con el nuevo elemento
10. Finalmente, colocamos fin en el nuevo elemento.

Algoritmo para llenar una lista enlazada simple

```
    public void agregar(){
1.  Dato d = new Dato();
2.  d.setIdProducto(Integer.parseInt(JOptionPane.
            showInputDialog(null, "Digite el id. del producto:")));
    d.setDescripcion(JOptionPane.showInputDialog(null,
            "Digite la descripción del producto:"));
3.  Nodo nuevo = new Nodo();
4.  nuevo.setElemento(d);
5.  if (esVacia()) {
6.      inicio = nuevo;
7.  } else if (d.getIdProducto() <
    inicio.getElemento().getIdProducto()) {
8.          nuevo.setSiguiente(inicio);
9.          inicio = nuevo;
10. } else if (inicio.getSiguiente() == null) {
11.          inicio.setSiguiente(nuevo);
12. } else {
13.          Nodo aux = inicio;
14.          while ((aux.getSiguiente() != null) &&
            (aux.getSiguiente().getElemento().getIdProducto()
            < d.getIdProducto())) {
                aux = aux.getSiguiente();
15.          }
16.     nuevo.setSiguiente(aux.getSiguiente());
17.     aux.setSiguiente(nuevo);
18.   }
    }
```

1. Se crea el objeto.
2. Se llena el objeto.
3. Se crea el nodo nuevo.
4. Se almacena el objeto en el nodo.
5. Preguntar si está vacía.
6. Si está vacía, colocamos el inicio en el nuevo nodo.
7. Si no, si el id del producto que voy a insertar es menor que el del inicio,
8. Colocamos nuevo de tal forma que el siguiente de él sea el nodo de inicio.
9. Colocamos inicio en el nuevo nodo.
10. Si vamos a insertar el nuevo nodo después del nodo de inicio entonces
11. Colocamos inicio de tal forma que el siguiente a él sea el nuevo elemento.
12. Si vamos a insertar en medio o al final, entonces
13. Creamos una variable de tipo nodo y la colocamos en inicio y
14. Recorremos la lista mientras el elemento siguiente de aux sea diferente de null y que el id del elemento siguiente de aux sea menor que el id del elemento que voy a insertar.
15. Mientras eso sucede, avanzamos aux al elemento siguiente para ubicarnos donde vamos a insertar el nuevo nodo. Y una vez ubicados…
16. Colocamos nuevo como el siguiente de aux y finalmente….
17. Enlazamos aux con nuevo.

Algoritmo para llenar una lista simple circular

```
public void agregar(){
1.  Persona p=new Persona();
2.  p.setId(Short.parseShort(JOptionPane.showInputDialog(null,
    "Digite el id.:")));
    p.setNomb(JOptionPane.showInputDialog(null,
    "Digite el nombre:"));
3.  Nodo nuevo=new Nodo();
4.  nuevo.setDato(p);
5.  if(esVacia()){
6.    inicio=nuevo;
7.    fin=inicio;
8.    fin.setSiguiente(inicio);
9.  }else if(p.getId()<inicio.getDato().getId()){
10.   nuevo.setSiguiente(inicio);
11.   inicio=nuevo;
12.    fin.setSiguiente(inicio);
13. }else if(p.getId()>=fin.getDato().getId()){
14.    fin.setSiguiente(nuevo);
15.    fin=fin.getSiguiente();
16.    fin.setSiguiente(inicio);
17. }else{
18.    Nodo aux=inicio;
19.    while(aux.getSiguiente().getDato().getId()<p.getId()){
20.         aux=aux.getSiguiente();
21.  }nuevo.setSiguiente(aux.getSiguiente());
22.  aux.setSiguiente(nuevo);
23.  fin.setSiguiente(inicio);
24. }
 }
```

1. Se crea el objeto.
2. Se llena el objeto.
3. Se crea el nodo nuevo.
4. Se almacena el objeto en el nodo.
5. Preguntar si está vacía.
6. Si está vacía, colocamos el inicio en el nuevo nodo.
7. Colocamos fin en el inicio.
8. Enlazamos fin con el inicio.
9. Si no, si el id del producto que voy a insertar es menor que el del inicio,
10. Colocamos nuevo de tal forma que el siguiente de él sea el nodo de inicio.
11. Colocamos inicio en el nuevo nodo.
12. Enlazamos fin con inicio.
13. Si el nodo que vamos a insertar tiene un id mayor o igual que el del final, entonces…
14. Colocamos fin de tal forma que el siguiente a él sea el nuevo elemento.
15. Colocamos fin en el nuevo elemento.
16. Enlazamos fin con inicio.
17. Si no, si vamos a insertar en medio o al final, entonces
18. Creamos una variable de tipo nodo y la colocamos en inicio y
19. Recorremos la lista mientras el id del elemento siguiente de aux sea menor que el id del elemento que voy a insertar.
20. Mientras eso sucede, avanzamos aux al elemento siguiente para ubicarnos donde vamos a insertar el nuevo nodo. Y una vez ubicados…
21. Colocamos nuevo como el siguiente de aux y finalmente….
22. Enlazamos aux con nuevo.
23. Y finalmente… enlazamos el fin con el inicio de la lista.

Algoritmo para llenar una lista doble circular

```
public void agregar(){
1.  Vehiculo v=new Vehiculo();
2.  v.setIdVehiculo(Integer.parseInt(JOptionPane.
    showInputDialog(null,"Digite el id. del vehículo:")));
    v.setMarca(JOptionPane.showInputDialog(null,
    "Digite la marca:"));
3.  NodoDC nuevo=new NodoDC();
4.  nuevo.setDato(v);
5.  if(esVacia()){
6.     inicio=nuevo;
7.     fin=nuevo;
8.     fin.setSiguiente(inicio);
9.     inicio.setAnterior(fin);
10. }else if(v.getIdVehiculo()<inicio.getDato().getIdVehiculo()){
11.    nuevo.setSiguiente(inicio);
12.    inicio=nuevo;
13.    fin.setSiguiente(inicio);
14.    inicio.setAnterior(fin);
15. }else if(v.getIdVehiculo()>=fin.getDato().getIdVehiculo()){
16.    fin.setSiguiente(nuevo);
17.    fin=fin.getSiguiente();  //fin=nuevo;
18.    fin.setSiguiente(inicio);
19.    inicio.setAnterior(fin);
20. }else{
21.    NodoDC aux=inicio;
22.    while(aux.getSiguiente().getDato().getIdVehiculo()<
           v.getIdVehiculo()){
23.       aux=aux.getSiguiente();
       }
24.   nuevo.setSiguiente(aux.getSiguiente());
25.   nuevo.setAnterior(aux);
26.   aux.setSiguiente(nuevo);
27.   nuevo.getSiguiente().setAnterior(nuevo);
28. }
 }
```

1. Se crea el objeto.
2. Se llena el objeto.
3. Se crea el nodo nuevo.
4. Se almacena el objeto en el nodo.
5. Preguntar si está vacía.
6. Si está vacía, colocamos el inicio en el nuevo nodo.
7. Colocamos fin en el inicio.
8. Enlazamos fin con el inicio.
9. Enlazamos inicio con el fin.
10. Si no, si el id del vehículo que voy a insertar es menor que el del inicio,
11. Colocamos nuevo de tal forma que el siguiente de él sea el nodo de inicio.
12. Colocamos inicio en el nuevo nodo.
13. Enlazamos fin con inicio.
14. Enlazamos inicio con fin.
15. Si el nodo que vamos a insertar tiene un id mayor o igual que el del final, entonces…
16. Colocamos fin de tal forma que el siguiente a él sea el nuevo elemento.
17. Colocamos fin en el nuevo elemento.
18. Enlazamos fin con inicio.
19. Enlazamos inicio con fin.
20. Si no, si vamos a insertar en medio o al final, entonces
21. Creamos una variable de tipo nodo y la colocamos en inicio y
22. Recorremos la lista mientras el id del elemento siguiente de aux sea menor que el id del elemento que voy a insertar.
23. Mientras eso sucede, avanzamos aux al elemento siguiente para ubicarnos donde vamos a insertar el nuevo nodo. Y una vez ubicados…
24. Colocamos nuevo como el siguiente de aux.
25. Enlazamos nuevo en su parte anterior con aux.
26. Enlazamos aux con nuevo.
27. Y finalmente… enlazamos el elemento siguiente de nuevo en su parte anterior con nuevo.

Algoritmo para llenar un árbol

```
public void insertar(int num) {
1.  if (raiz == null) {
2.  raiz = new NodoArbol(num);
3.  } else {
4.  inserta(raiz, num);
5.  }
    }

7.  public void inserta(NodoArbol n, int num) {
8.  if (num <= n.getNum()) {
9.      if (n.getHijoIzq() == null) {
10.         n.setHijoIzq(new NodoArbol(num));
11.     } else {
            inserta(n.getHijoIzq(), num);
     }
12. } else {
13.    if (n.getHijoDer() == null) {
14.         n.setHijoDer(new NodoArbol(num));
15.    } else {
16.       inserta(n.getHijoDer(), num);
17.    }
     }
    }
```

1. Si la raíz está vacía.
2. La creo y almaceno allí el nuevo elemento.
3. Si no…
4. Me ubico recursivamente en el nodo respectivo llamo al método inserta y ahí determino si…
8. Si el número es menor va a la izquierda, entonces…
9. Reviso si a la izquierda está vacío.
10. Si sí, entonces creo el nuevo nodo ahí, sino…
11. Me ubico recursivamente en el nodo respectivo para insertar el nuevo valor.
12. Si no… (Si es mayor y va a la derecha)
13. Reviso si a la derecha está vacío.
14. Si sí agrego ahí el nuevo nodo.
15. Si no…
16. Me ubico recursivamente en el nodo respectivo y allí agregro el nuevo valor.