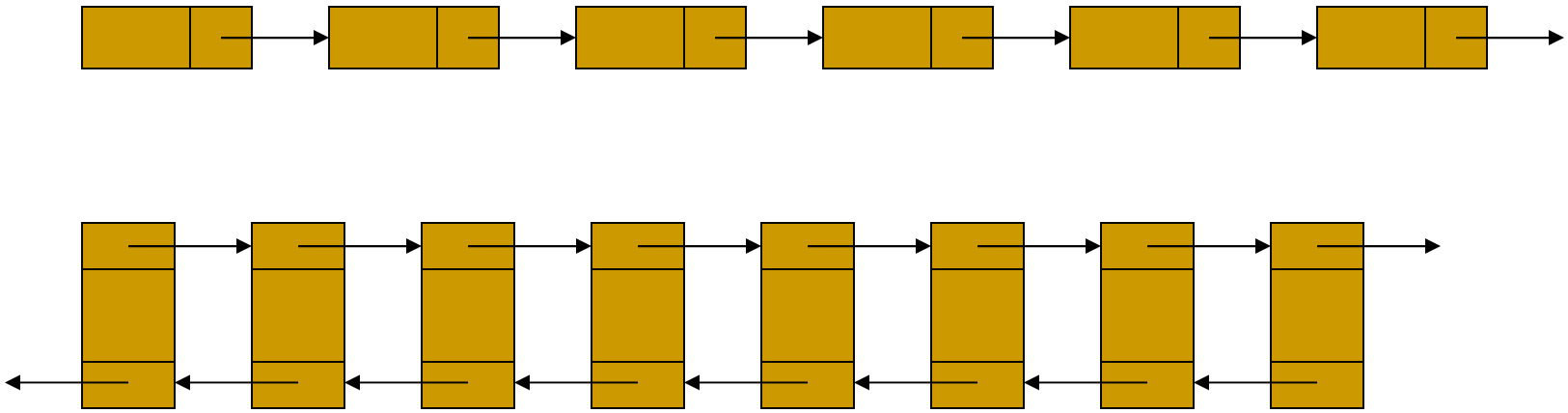
Danh sách liên kết (Linked List)

Giới thiệu

- Định nghĩa: Là danh sách bao gồm các phần tử kết nối với nhau bằng 1 hay nhiều mối liên kết

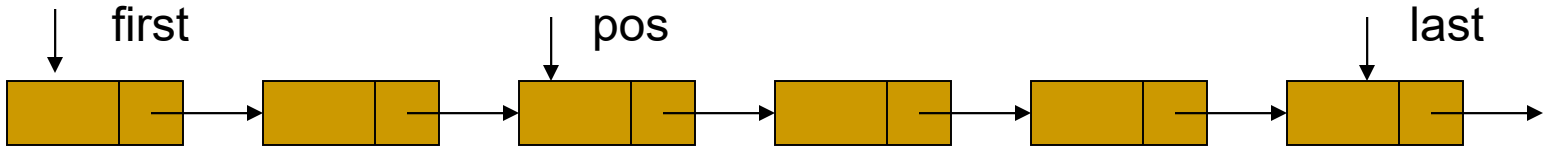


Các thao tác trên danh sách

- Khởi tạo danh sách rỗng
- Kiểm tra danh sách rỗng?
- Chèn phần tử vào danh sách
- Xóa phần tử trong danh sách
- Tìm kiếm phần tử trong danh sách
- Khởi đầu từ đầu danh sách
- Lấy dữ liệu 1 phần tử
- Chuyển sang phần tử kế tiếp
- Kiểm tra hết danh sách
- ….

Danh sách liên kết đơn

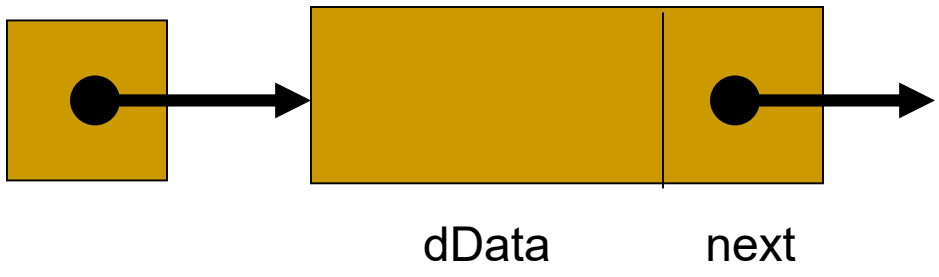
- Là danh sách mà mỗi phần tử có 1 mối liên kết để kết nối với phần tử kế tiếp



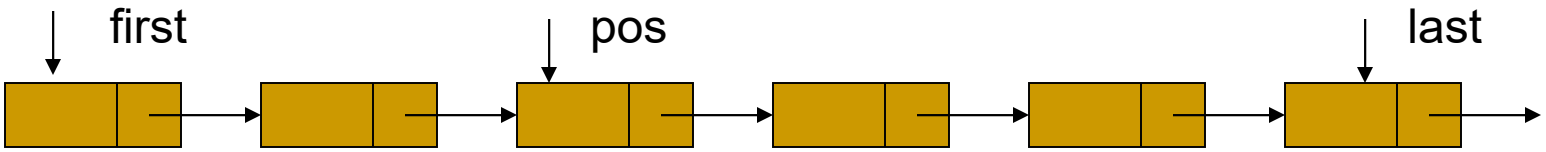
- Cài đặt: dựa trên tham chiếu, bao gồm:
  - 3 object: first (đầu ds), pos (phần tử hiện hành), và last (cuối ds)
  - biến count: số phần tử của danh sách

Mô tả kiểu dữ liệu

```
class Link
{
     public int dData;          // data item
     public Link next;          // next link in list
    public Link(int d)          // constructor
    {
       dData = d;
    }
}
```



Mô tả kiểu dữ liệu




```
class LinkedList
{
    private Link first;          // ref to first link
    private Link last;           // ref to last link
    private Link pos;            // ref to last link
    private int count;
        //----  Các method ----
}
```

Khởi tạo danh sách rỗng

Gán

- first, pos và last = null
- count = 0

```
public LinkedList()                 // constructor
{
        first = null;               // no links on list yet
        last = null;
        pos = null;
        count = 0;
}
```

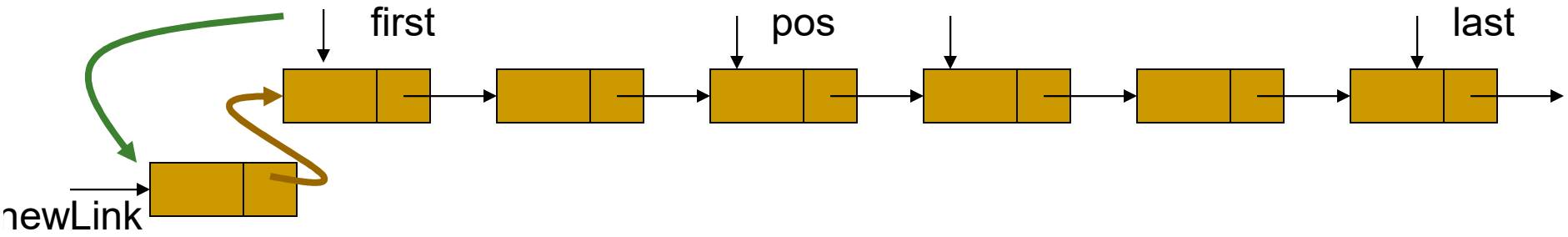
Kiểm tra danh sách rỗng?

- Kiểm tra số phần tử = 0, hay first = null

```
public boolean isEmpty()     // true if no links
{
        return first==null;
}
```

Chèn phần tử vào danh sách – local

- Cấp phát bộ nhớ cho newLink (và gán dữ liệu)
- Chèn ở đầu ds *(p==null)*

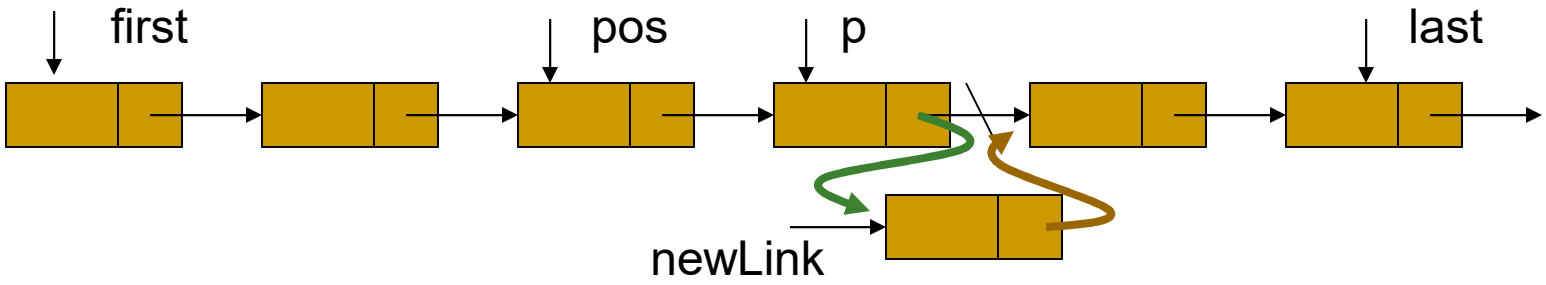


newLink.next = first;

first = newLink;

Chèn phần tử vào danh sách – local

- Chèn sau phần tử p *(p!=null)*

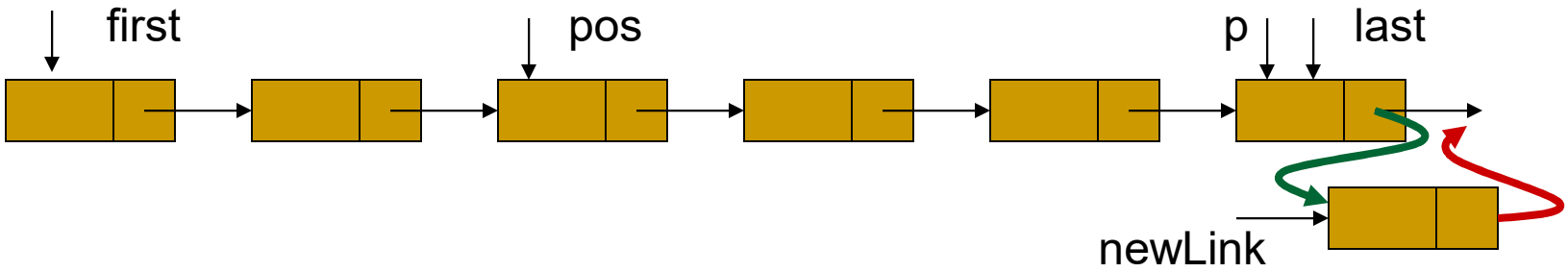


newLink.next = p.next;

p.next = newLink;

Chèn phần tử vào danh sách – local

- Trường hợp chèn cuối (newLink.next == null)



- Thay đổi *last*

last = *newLink*;

Chèn phần tử vào danh sách – private

```
private void insert(int dd, Link p)                    // local
{
      Link newLink = new Link(dd);     // make new link
      if (p == null)
      {
            newLink.next =first;
            first = newLink;
      }
      else
      {
            newLink.next = p.next;
            p.next = newLink;
      }
      if (newLink.next == null)
            last = newLink;
      count++;
}
```

Các hàm chèn phần tử – public

- Chèn đầu danh sách

```
public void insertFirst(int dd)        // insert at front of list
{
    insert(dd, null);
}
```

- Chèn vị trí hiện hành

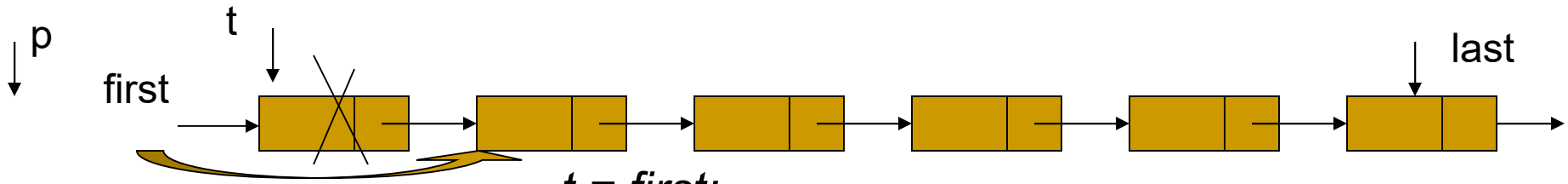
```
public void insertPos(int dd)
{
    insert(dd, pos);
}
```

- Chèn cuối danh sách

```
public void insertLast(int dd)         // insert at end of list
{
    insert(dd, last);
}
```

Xoá phần tử trong danh sách – local

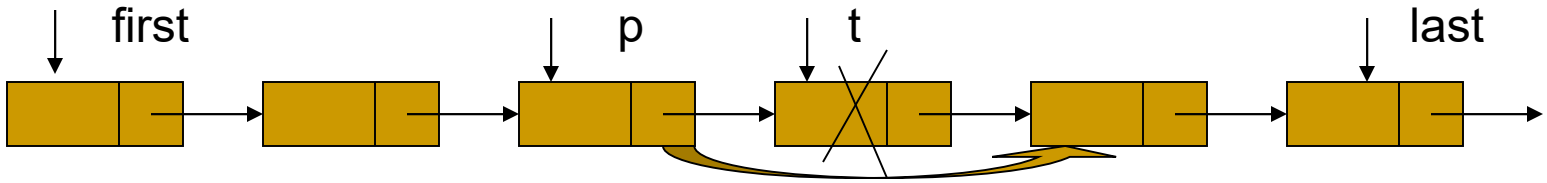
- Xoá phần tử đầu danh sách (p=null)



t = first;

first = t.next;

- Xoá phần tử sau p (p!=null)



t = p.next;

p.next = t.next;

```
private void delete(Link p)
{
        Link t;
        if (p==null)
        {
                t = first;
                first = t.next;
        }
        else
        {
                t = p.next;
                p.next = t.next;
        }
        if (t.next == null)
                last = p;
        //t = null;
        count--;
}
```

Các hàm xóa phần tử – public

- Xóa phần tử đầu danh sách

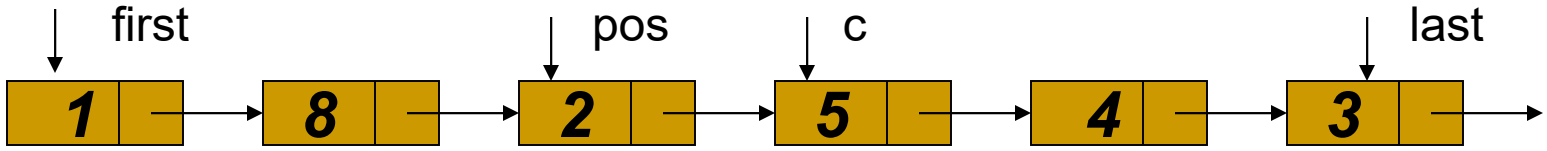
```
public void deleteFirst()                    // delete first link
{
        delete(null);
}
```

- Xóa phần tử sau vị trí hiện hành

```
public void deletePos() // delete first link
{
        delete(pos);
}
```

Tìm kiếm trên DS

dd=5



1. *Khởi đầu từ đầu danh sách (pos = null; c = first)*
2. *Khi chưa hết ds và chưa tìm thấy*

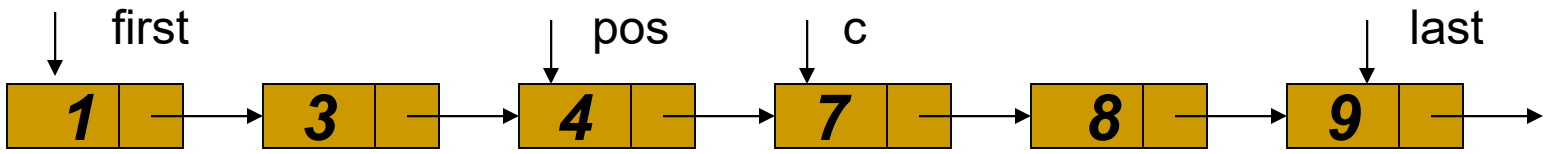
   Chuyển sang phần tử kế tiếp (pos = c; c = c.next)

Tìm kiếm trên DS ngẫu nhiên

```
public boolean searchList(int dd)
{
    Link  c;
    c = first;
    pos = null;
    while (c!=null && c.dData!=dd)
    {
        pos = c;
        c = c.next;
    }
    return c!=null;
}
```

Tìm kiếm trên DS có thứ tự

dd=5



1. *Khởi đầu từ đầu danh sách (pos = null; c = first)*
2. *Khi chưa hết ds và dd>c.dData*

   Chuyển sang phần tử kế tiếp (pos = c; c = c.next)

Tìm kiếm trên DS có thứ tự

```
public boolean searchOrderList(int dd)
{
    Link  c;
    c = first;
    pos = null;
    while (c!=null && c.dData<dd)
    {
        pos = c;
        c = c.next;
    }
    if (c!=null && c.dData>dd)
        return false;
    return c!=null;
}
```

Một số hàm hỗ trợ

- Khởi đầu từ đầu danh sách

```
public void startList()
{
        pos = first;
}
```

- Chuyển sang phần tử kế tiếp

```
public void nextLink()
{
      if (pos == null)
            pos = first;
      pos = pos.next;
}
```

- Kiểm tra hết danh sách

```
public boolean endList()
{
    return  pos == null;
}
```

- Lấy dữ liệu 1 phần tử

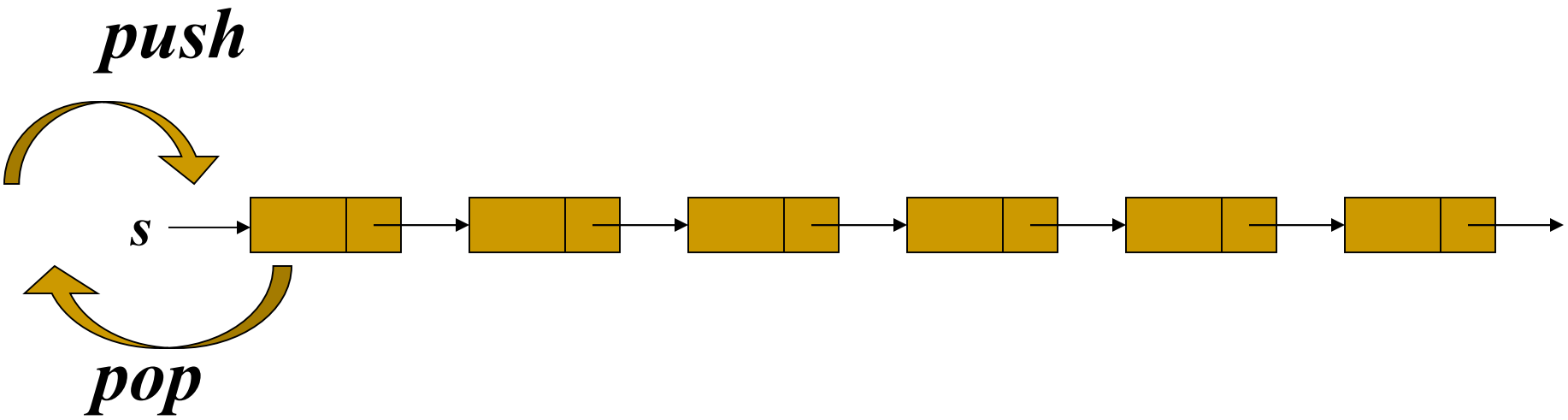
```
public int getData()
{
    return  pos.dData;
}
```

- Số phần tử trong danh sách

```
public int nItem()      //number of item
{
    return  count;
}
```

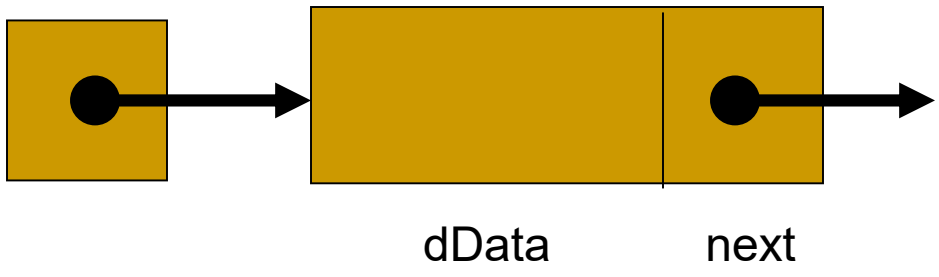
Stack- Cài đặt trên cơ sở liên kết

- Cài đặt tương tự như danh sách liên kết đơn với 2 thao tác
  - Chèn đầu danh sách
  - Lấy ra ở đầu danh sách

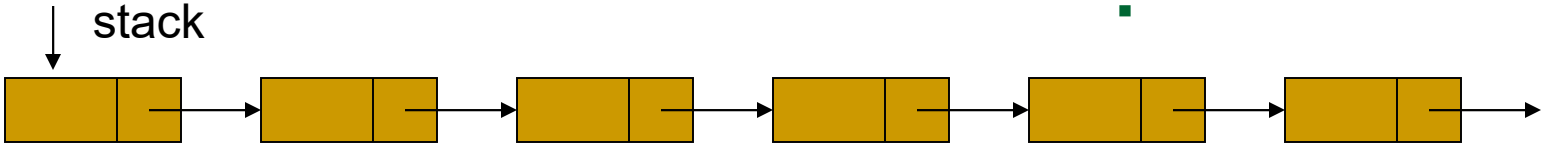


Mô tả kiểu dữ liệu

```
class Link
{
     public int dData;          // data item
     public Link next;          // next link in list
    public Link(int d)          // constructor
    {
       dData = d;
    }
}
```



Mô tả kiểu dữ liệu

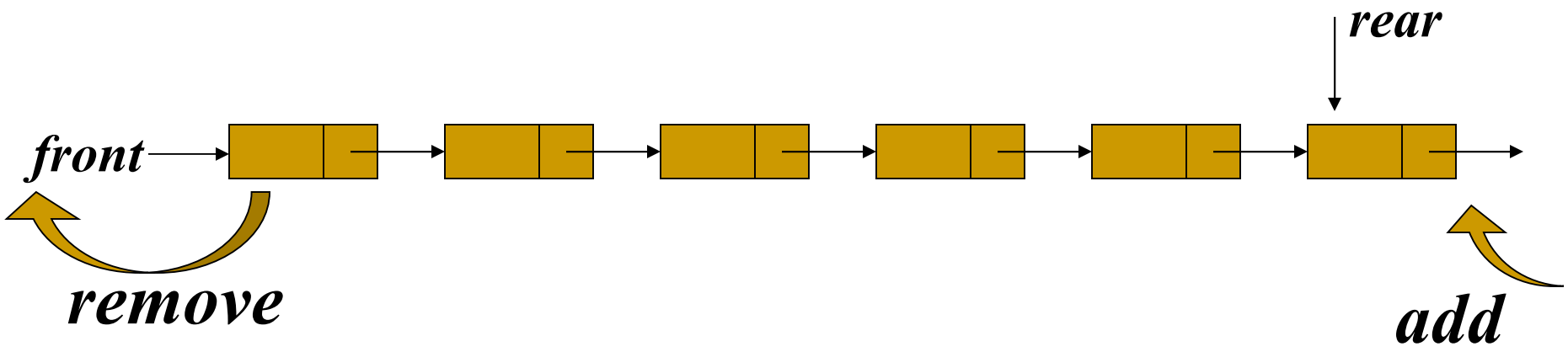



```
class Stack
{
   private Link first; // ref to first link
   private int count;
   public Stack() // constructor
   {
      first = null; // no links on list yet
      count = 0;
   }
   public boolean isEmpty() // true if no links
   {
      return first==null;
   }
```

```
  public void push(int dd) {
        Link newLink = new Link(dd); // make new link
        newLink.next =first;
        first = newLink;
        count++;
   }
   public int pop()
  {
        Link t;
        t = first;
        first = t.next;
        count--;
        return t.dData;
   }
}
```

Queue - Cài đặt trên cơ sở con trỏ

- Cài đặt tương tự như danh sách liên kết đơn với 2 thao tác
  - Chèn cuối danh sách
  - Lấy ra ở đầu danh sách



- Viết ***hàm*** tách danh sách liên kết đơn chứa các số nguyên thành 2 danh sách chẵn/lẻ
- Viết hàm trộn 2 danh sách có thứ tự thành 1 danh sách có thứ tự
- Sắp thứ tự DSLK bằng phương pháp ***chèn***
- Sắp thứ tự DSLK bằng phương pháp ***chọn***
- Sắp thứ tự DSLK bằng phương pháp ***trộn***

```
public void insertionsort()
{
    Link c, h, p, q, r;
    h = r = new Link(0);// cap phat phan tu dem
    h.next = null;
    q = first;          // Xet tu dau ds
    while (q != null)
    {
        p = h;
        c = h.next;
        while (c!=null && q.dData > c.dData)
        {
            p = c;
            c = c.next;
        }
        p.next = q;
        p = q;
        q = q.next;
        p.next = c;
        if (c==null)
            r = p;
    }
    first = h.next;
    last = r;
}
```

```
public void selectionsort()    //Phương pháp chọn
{
    Link c, cm, p, pm, q, r;
    q = new Link(0);
    q.next = first;
    r = null;
    while (q.next!=null)
    {
        pm = p = q;
        cm = c = q.next;
        while (c!=null)
        {
            if (cm.dData < c.dData)
            {
                pm = p;
                cm = c;
            }
            p = c;
            c = c.next;
        }
        if (r==null)
            last = cm;
        pm.next = cm.next;
        cm.next = r;
        r = cm;
    }
    first = r;
}
```

- Xin vui lòng yên lặng!