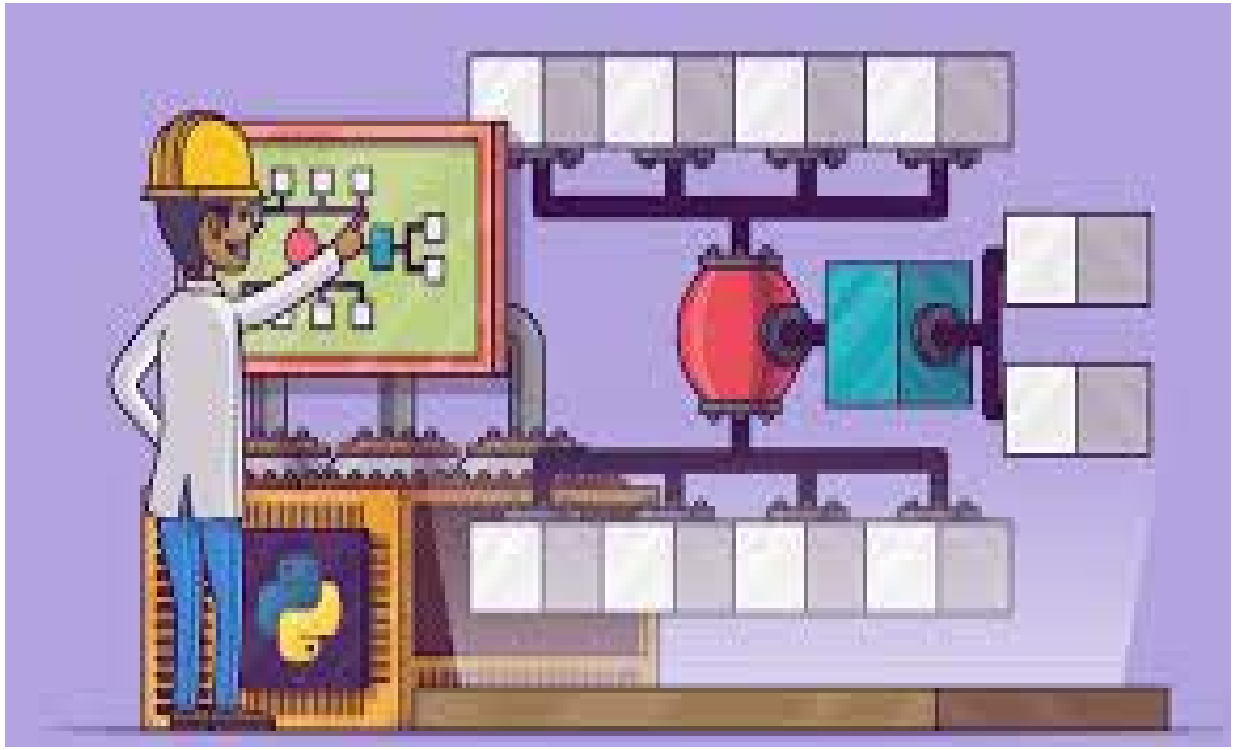
ساختمان داده ها



مدرس:
سمانه حسینی سمنانی

دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر

Linked List لیست پیوندی

- لیست های یک پیوندی و زنجیرها
- لیست های حلقوی
- پشته ها و صف های پیوندی
- چند جمله ای ها
- لیستهای دو پیوندی

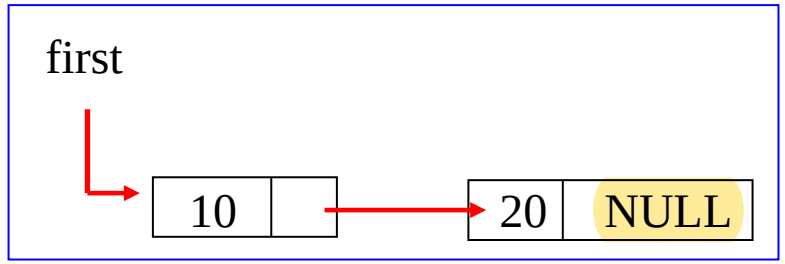
عملیات روی زنجیرها

```
class ChainNode {
friend class Chain;
public:
  ChainNode (int element = 0, ChainNode* next = 0)
        // 0 is the default value for element and next
        {data = element; link = next;}
private:
  int data;
  ChainNode *link;
```

- کلاس chain
- private:
  - اشاره گر first
- public:
  - تابع chain::Creat2
  - تابع chain::Insert
  - تابع chain::Delete
  - تابع chain::InsertBack
  - تابع chain::Concatenate
  - تابع chain::Revers

chain::Creat2

```
ChainNode chain::Create2(){
    /* create a linked list with two nodes */
        ChainNode *first, *second;
        first = new ChainNode;
        secound = new ChainNode;
        second -> link = NULL;
        second -> data = 20;
        first -> data = 10;
        first ->link = second;
        return first;
}
```

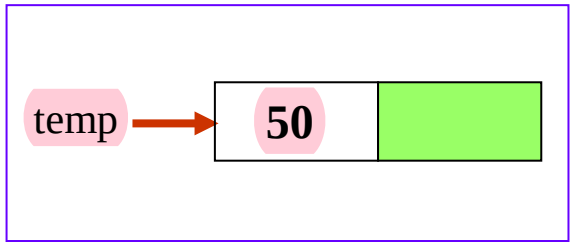


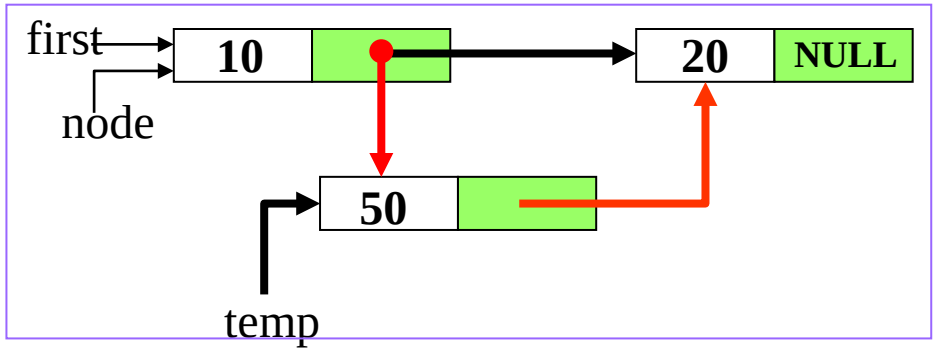
پیچیدگی O(1)

chain::Insert

```
void chain::Insert(ChainNode *temp, ChainNode *node){
      if(*first){     //nonempty list
            temp->link = node->link;
            node->link = temp;
          }
      else{           //empty list
            temp->link = NULL;
            *first = temp;
          }
      }

}
```



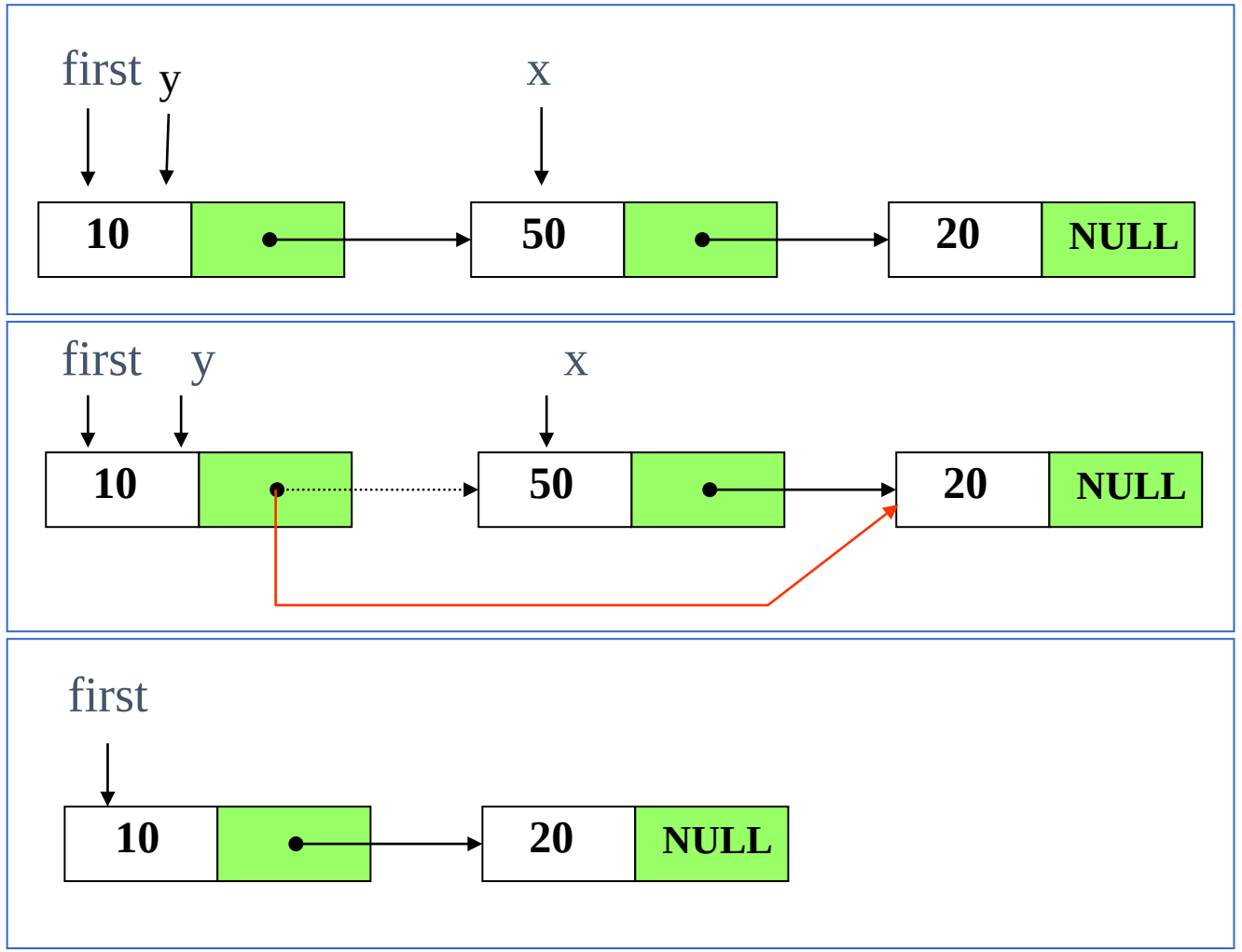


پیچیدگی O(1)

chain::Delete

```
void Chain::Delete(ChainNode *x, ChainNode *y)
{
    if (x == first) first = first → link;
    else y → link = x → link;
    delete x;
}
```

پیچیدگی O(1)



سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

chain::InsertBack

- اضافه کردن یک عضو در انتهای یک زنجیر

```
template <class T >
void Chain <T >::InsertBack(const T& e)
{
    if (first) {// nonempty chain
        last → link = new ChainNode <T >(e);
        last = last→link;
    }
    else first = last = new ChainNode<T >(e);
}
```

پیچیدگی O(1) به شرط نگهداری اشاره گر last
در غیر این صورت O(n)

chain::Concatenate

- اتصال دو زنجیر

```
template <class T>
void Cahin <T>::Concatenate(Chain<T>& b)
    {// b is concatenated to the end of *this.
if (first) (last →link = b.first; last = b.last;}
    else { first = b.first; last = b.last;}
    b.first = b.last = 0;
}
```

پیچیدگی O(1) به شرط نگهداری اشاره گر last
در غیر این صورت برای لیست با n عضو O(n)

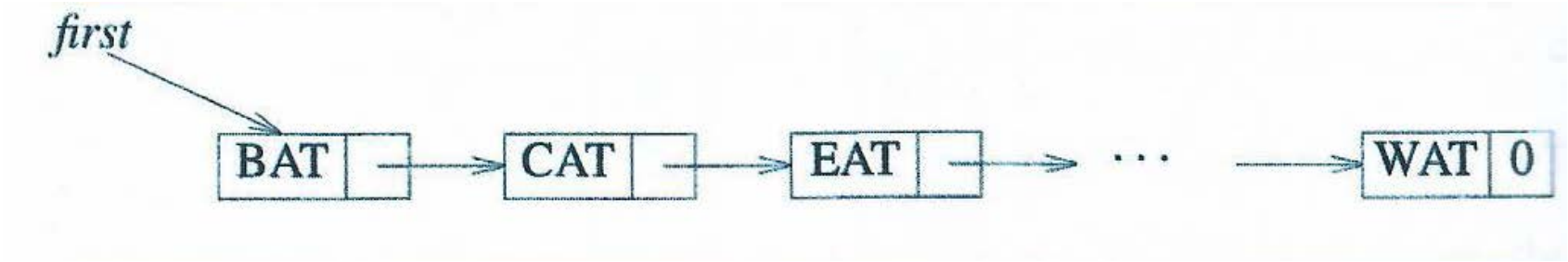
chain::Revers

- معکوس کردن یک زنجیر

```
template <class T >
void Cahin <T >::Revers ()
{// A chain is reserved so that (a1,..., an) becomes (an,..., a1).
      ChainNode<T > *current = first,
                  *previous = 0; // previous trails current
      while (current) {
            ChainNode  <T > *r = previous;
            previous = current;       // r trails previous
            current = current → link;     // current move to next node
            previous → link = r;      // link previous to preceding node
      }
      first = previous;
}
```

پیچیدگی O(n)

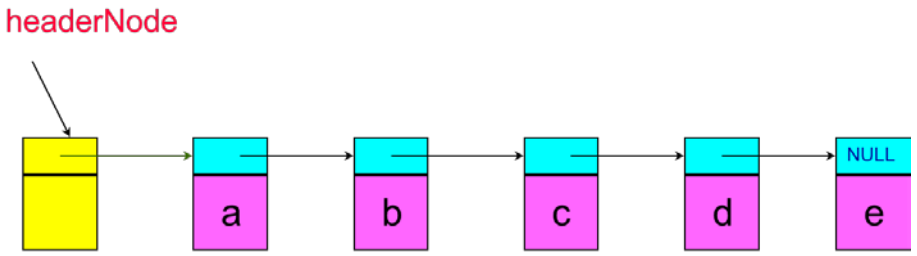
محاسبه طول زنجیر



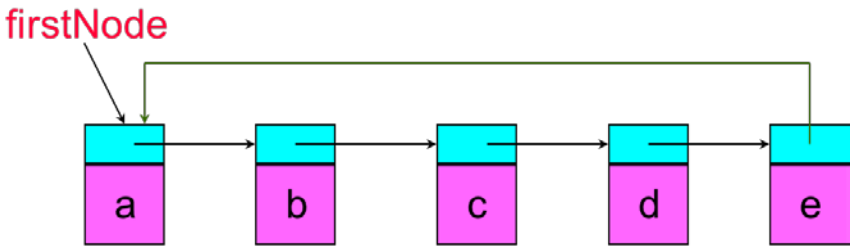
- پیدا کردن سایز لیست پیچیدگی O(n)
- می توان با اضافه کردن یک متغیر size که با هر بار حذف و اضافه به روز می شود این پیچیدگی را تبدیل به O(1) کرد
- Tradeoff بین زمان اجرا و حافظه اختصاصی

لیست های حلقوی

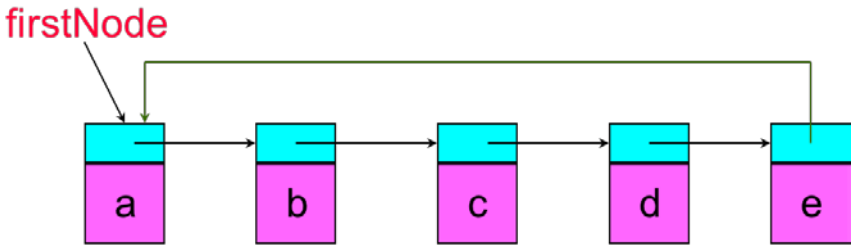
- زنجیر



- لیست حلقوی

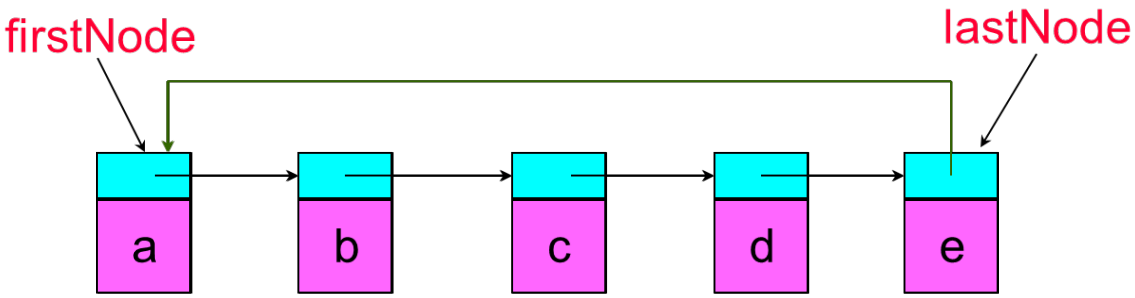


لیست های حلقوی



- بررسی اینکه آیا اشاره گر current به گره آخر لیست دایره ای اشاره می کند:
  - current->link==0 در زنجیرها ⟵ current->link==first در لیست حلقوی
- توابع حذف و اضافه باید حلقوی بودن لیست را حفظ کنند

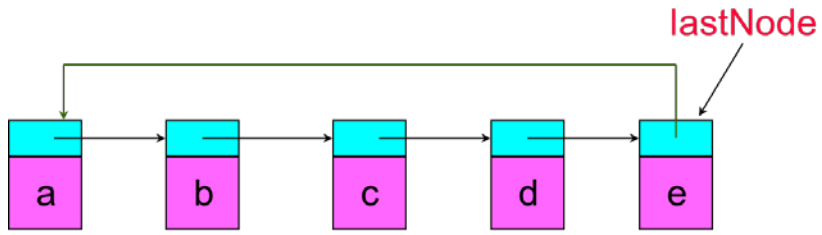
اضافه کردن گره جدید به اول لیست



- اشاره گر firstNode
- اشاره گر lastNode ←
- ساده تر شدن عملیات

اضافه کردن گره جدید به اول لیست

```
template <classT >
void CircularList <T >::InsertFront(const T& e)
{// Insert the element e at the "front" of the circular
// list *this, where last points to the last node in the list.
      ChainNode<T > *newNode = new ChainNode<T > (e);
      if (last) { // nonempty list
            newNode → link = last → link;
            last → link = newNode;
      }
      else { // empty list
            last = newNode ;
            newNode → link = newNode;
      }
}
```

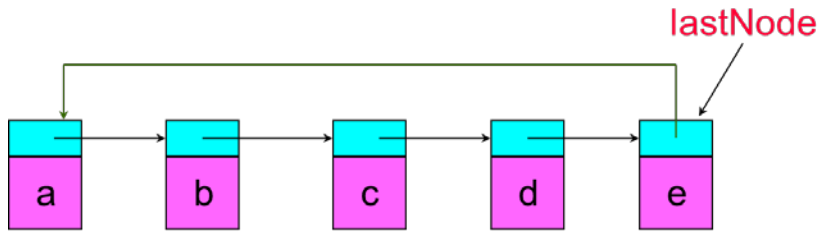


پیچیدگی O(1)

اضافه کردن گره جدید به آخر لیست

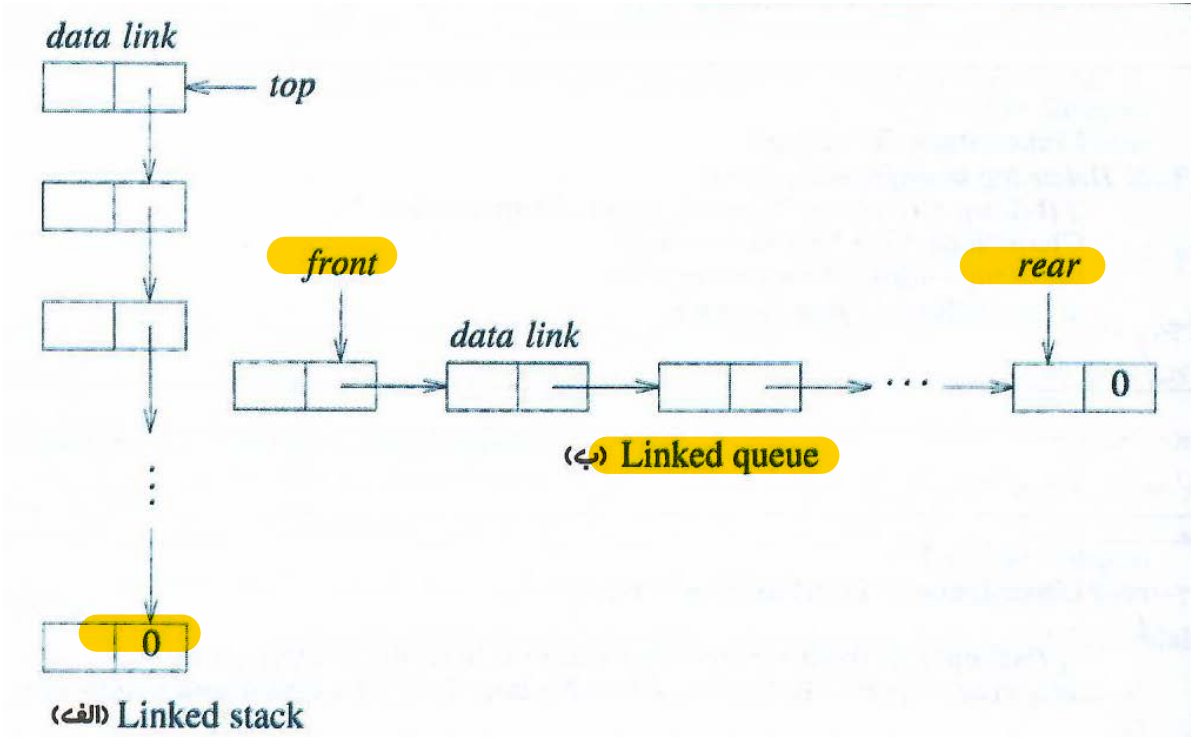
```
template <classT >
void CircularList <T >::InsertFront(const T& e)
{// Insert the element e at the "front" of the circular
// list *this, where last points to the last node in the list.
      ChainNode<T > *newNode = new ChainNode<T > (e);
      if (last) { // nonempty list
            newNode → link = last → link;
            last → link = newNode;
      }
      else { // empty list
            last = newNode ;
            newNode → link = newNode;
      }
}
```

last = newNode;



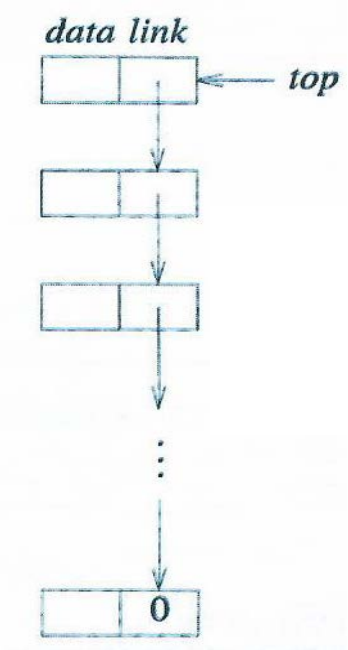
پیچیدگی O(1)

پشته و صف پیوندی



- جهت اشاره گر برای پشته و صف به صورتی است که عملیات حذف کردن و اضافه کردن گره ها در آنها به آسانی انجام شود.
- در شکل های روبرو دقت کنید:
  - به آسانی می توانید یک گره را به بالای پشته اضافه و یا از آن حذف کنید.
  - به آسانی می توانید یک گره به آخر صف اضافه کنید یا عمل اضافه کردن و حذف کردن را در اول صف انجام دهید (هر چند اضافه کردن گره در اول صف معمولا انجام نمی شود)

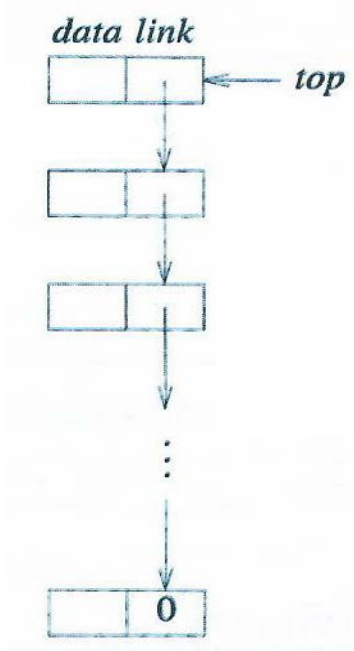
اضافه کردن یک گره در پشته پیوندی




```
template <class T >
void LinkedStack <T >::Push(const T& e) {
    top = new ChainNode<T >(e, top);
}
```

- تعریف دو کلاس LinkedStack و LinkedQueue
- هر دو friend کلاس ChainNode هستند.
- LinkedStack یک داده خصوصی Top دارد که به بالای استک اشاره می کند.
- LinkedQueue دو داده خصوصی front و rear دارد که به ترتیب به اول و آخر صف اشاره می کنند.
- سازنده کلاس LinkedStack مقدار top را برابر با null می گذارد.
- سازنده کلاس LinkedQueue مقدار front و rear را برابر با null می گذارد.

حذف یک گره در پشته پیوندی




```
template <class T >
void LinkedStack<T >::Pop()
{// Delete top node from the stack.
      if (IsEmpty ()) throw "Stack is empty. Cannot delete. ";
      ChainNode <T > *delNode = top;
      top = top →link; //remove top node
      delete delNode;// free the node
}
```