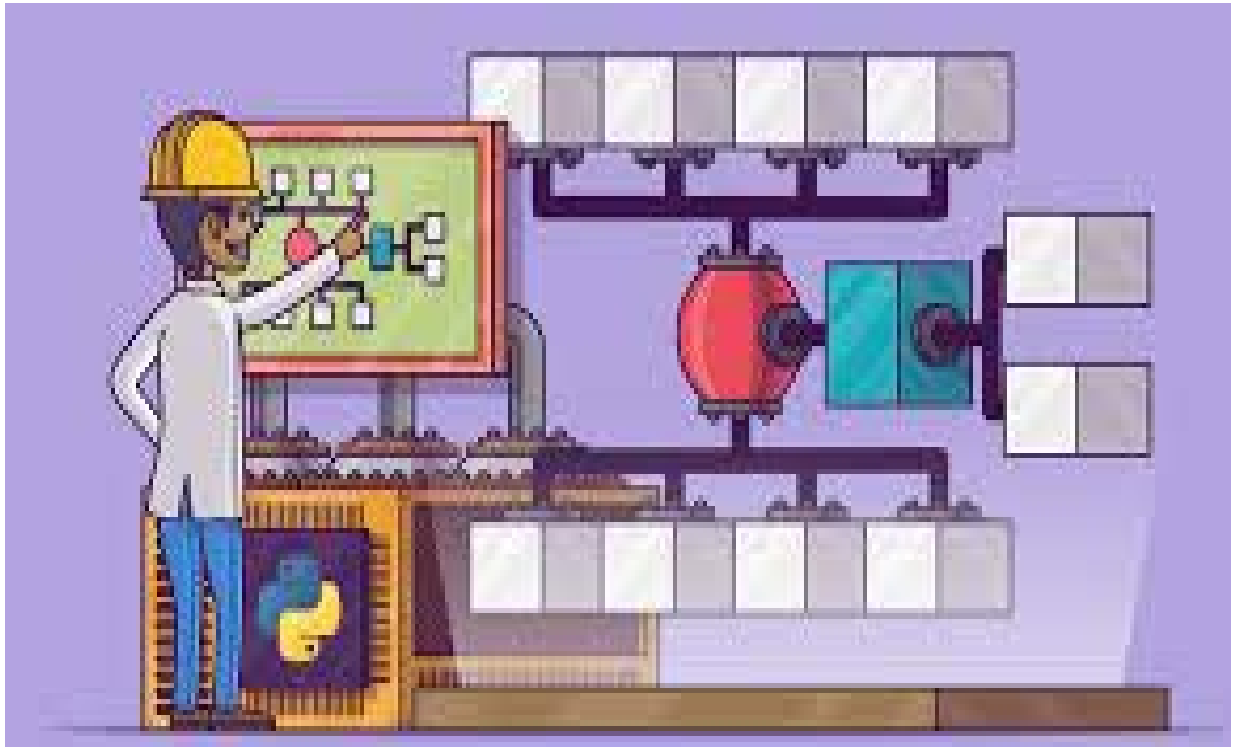
ساختمان داده ها



مدرس:
سمانه حسینی سمنانی

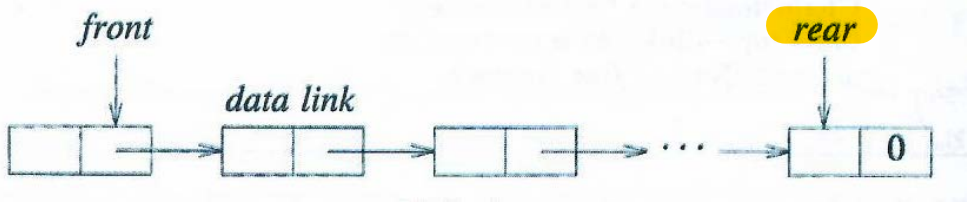
دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر

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

اضافه کردن یک گره در صف پیوندی

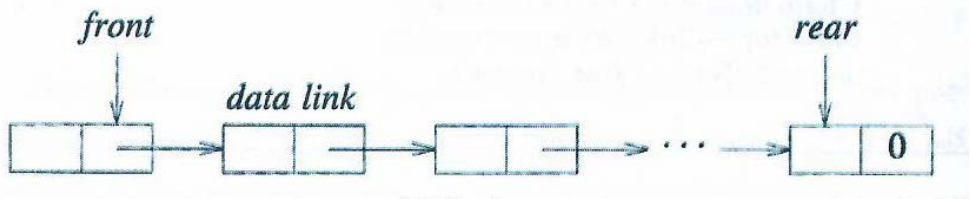
```
template <class T >
void LinkesQueue <T >::Push(const T& e)
{
    if (IsEmpty ()) front = rear = new ChainNode(e, 0); // empty queue
    else rear = rear → link = new ChainNode(e, 0); // attach node and update rear
}
```



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

حذف یک گره در صف پیوندی

```
template <class T >
void LinkedQueue <T >::Pop()
{// Delete first element in queue.
      if (IsEmpty ()) throw "Queue is empty. Cannot delete.";
      ChainNode<T > *delNode = front;
      front = front →link;       // remove first node from chain
      delete delNode;    // free the node
}
```



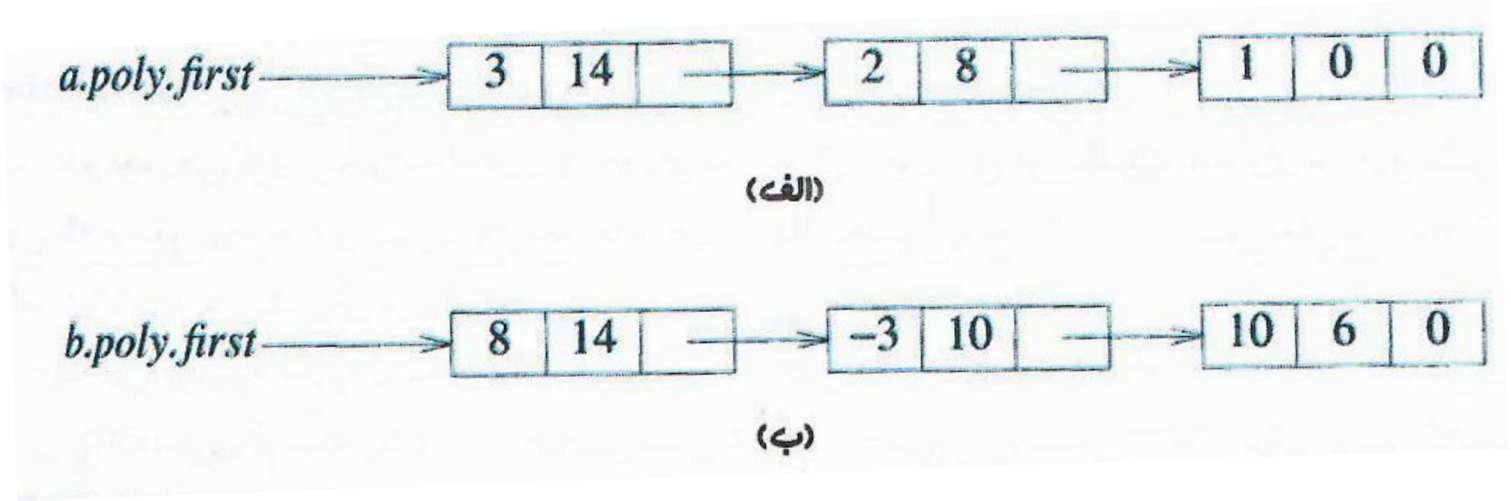
سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

پشته و صف پیوندی

- پشته و صف پیوندی نسبت به پشته و صف آرایه ای هم از نظر محاسباتی و هم از نظر مفهومی بهتر است:
  - لازم نیست برای ایجاد فضای خالی عناصر پشته یا صف شیفت داده شوند.
  - تا زمانی که حافظه وجود داشته باشد می توان از آن استفاده کرد.

نمایش چند جمله ای ها

$$a = 3x^{14} + 2x^{8} + 1 \qquad b = 8x^{14} - 3x^{10} + 10x^{6}$$

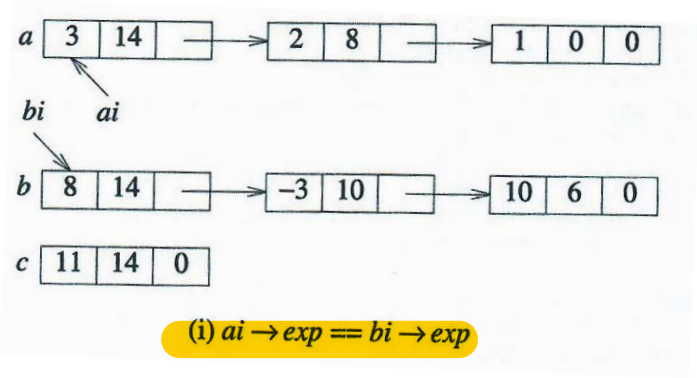


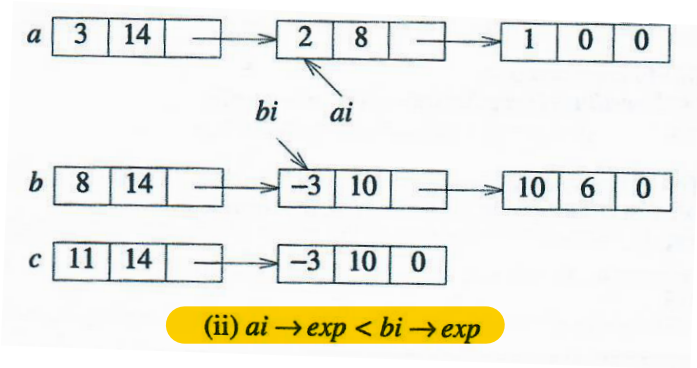
نمایش چند جمله ای ها

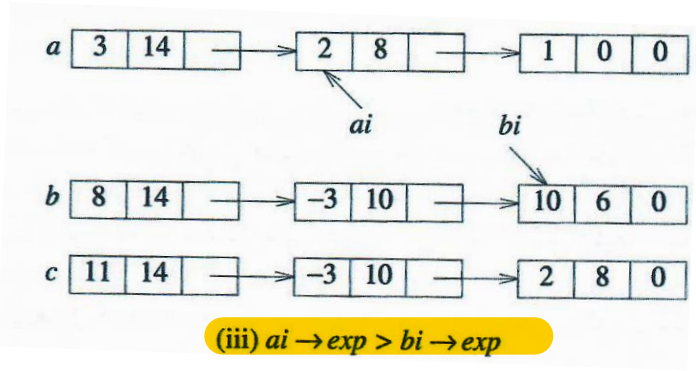
```
class Polynomial {
public:
    // public functions defined here
private:
    Chain<Term > poly;
};
```

```
struct Term
{// All members of Term are public by default.
    int coef; // coefficient
    int exp;  // exponent
    Term Set(int c, int e) {coef = c; exp = e; return *this;};
};
```

جمع چند جمله ای ها







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

جمع چند جمله ای ها

```
1  Polynomial polynomial::operator+(const Polynomial& b) const
2  {// Polynomial *this (a) and b are added and the sum returned.
3        Term temp;
4        Chain<Term>::ChainIterator ai = poly.begin (),
5                                   bi =b.poly.begin ();
6        Polynomial c;
7        while (ai && bi) { // current nodes are not null
8           if (ai →exp == bi→exp)) {
9              int sum = ai → coef + bi → coef;
10             if (sum) c.poly.InsertBack (temp.Set (sum, ai → exp));
11             ai ++; bi ++; // advance to next term
12          }
13          else if (ai →exp < bi →exp) {
14                c.poly.InsertBack(temp.Set (bi →coef , bi →exp);
15                bi ++; // next term of b
16          }
17          else {
18                c.poly.InsertBack (temp.Set (ai →coef , ai → exp));
19                ai ++; // next term of a
20           }
21       }
22       while (ai) {// copy rest of a
23          c.poly.InsertBack (temp.Set (ai →coef, ai →exp));
24          ai ++;
25       }
26       while (bi) {// copy rest of b
27          c.poly.InsertBack (temp.Set (bi →coef, bi →exp));
28          bi ++;
29       }
30       return c;
31 }
```


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

تحلیل پیچیدگی جمع چند جمله ای ها

```
1  Polynomial polynomial::operator+(const Polynomial& b) const
2  {// Polynomial *this (a) and b are added and the sum returned.
3        Term temp;
4        Chain<Term>::ChainIterator ai = pily.begin (),
5                                   bi =b.poly.begin ();
6        Polynomial c;
7        while (ai && bi) { // current nodes are not null
8           if (ai →exp == bi→exp)) {
9              int sum = ai → coef + bi → coef;
10             if (sum) c.poly.InsertBack (temp.Set (sum, ai → exp));
11             ai ++; bi ++; // advance to next term
12          }
13          else if (ai →exp < bi →exp) {
14                c.poly.InsertBack(temp.Set (bi →coef , bi →exp);
15                bi ++; // next term of b
16          }
17          else {
18                c.poly.InsertBack (temp.Set (ai →coef , ai → exp));
19                ai ++; // next term of a
20           }
21       }
22       while (ai) {// copy rest of a
23          c.poly.InsertBack (temp.Set (ai →coef, ai →exp));
24          ai ++;
25       }
26       while (bi) {// copy rest of b
27          c.poly.InsertBack (temp.Set (bi →coef, bi →exp));
28          bi ++;
29       }
30       return c;
31 }
```

$$A(x)(=a_{m-1}x^{e_{m-1}}+\cdots+a_0x^{e_0})+B(x)(=b_{n-1}x^{f_{n-1}}+\cdots+b_0x^{f_0})$$

- جمع ضرایب

$0 \le \text{additions} \le min(m, n)$

where m (n) denotes the number of terms in A (B)

- مقایسه توان ها

بدترین حالت:

$e_{m-1} > f_{m-1} > e_{m-2} > f_{m-2} > \ldots > e_1 > f_1 > e_0 > f_0$

$m+n-1$ comparisons

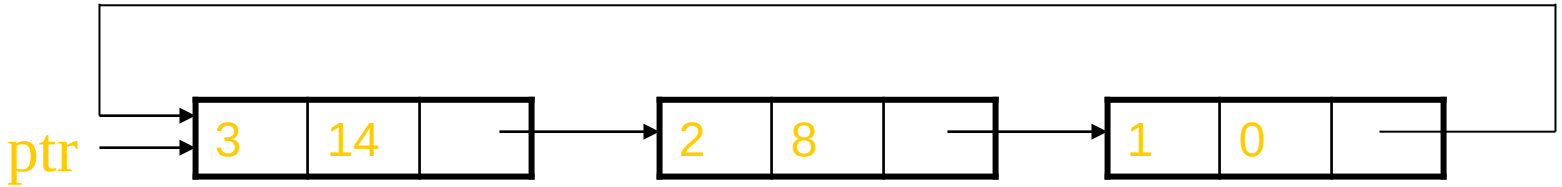
- ایجاد گره جدید برای c
  - بدترین حالت: m+n

پیچیدگی O(m+n)

الگوریتم بهینه

نمایش چند جمله ای با استفاده از لیست حلقوی

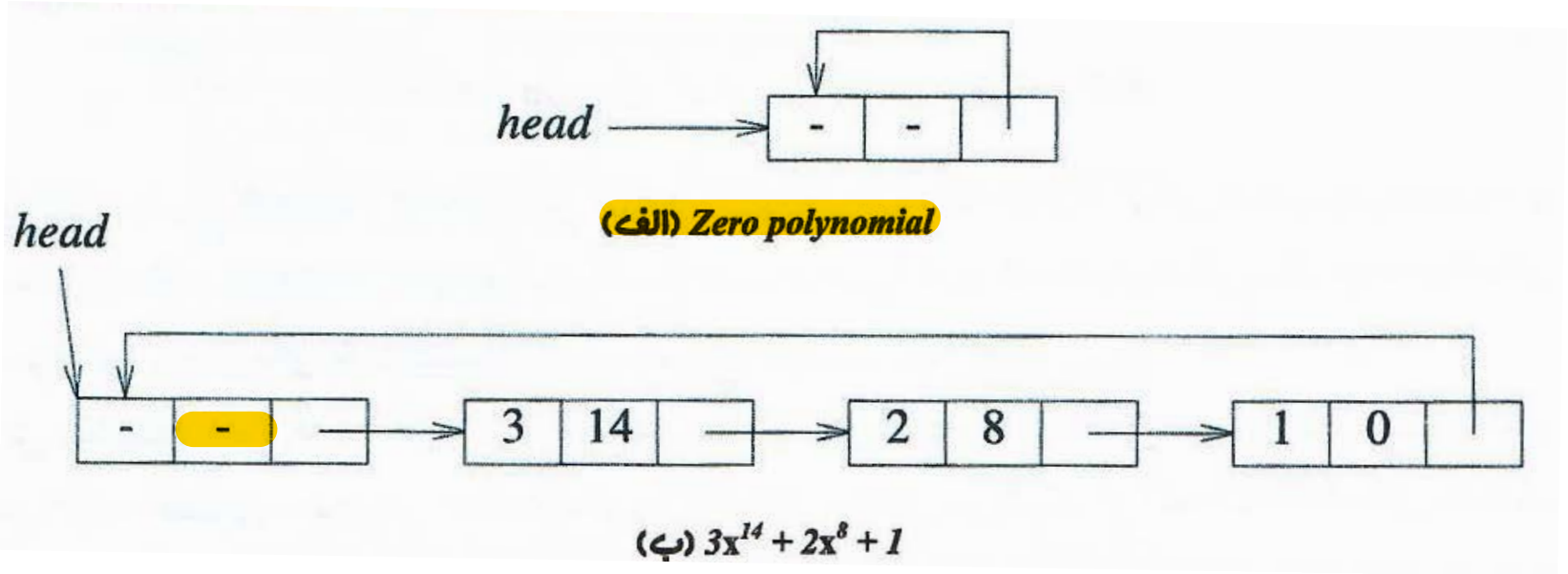
$$ptr = 3x^{14}+2x^{8}+1$$



- مشکل : باید چند جمله ای صفر را به صورت یک حالت خاص در نظر بگیریم . برای انجام عملیات جمع و ... دچار مشکل میشویم.

نمایش چند جمله ای با استفاده از لیست حلقوی

- راه حل : در نظر گرفتن یک گره سر اضافه
- عضوهای exp, coef بی اعتبار و بی معنا هستند در گره سر



جمع چند جمله ای با استفاده از لیست حلقوی

```
1  Polynomial Polynomial::operator+(const Polynomial& b) const
2  {// Polynomials *this (a) and b are added and the sum returned.
3        Term temp;
4        CircularListWithHeader<Term>::Iterator ai = poly.begin(),
5                                                bi = b.poly.begin();
6        Polynomial c; // assume constructor sets head →exp = -1
7        while (1) {
8           if (ai →exp == bi →exp)) {
9              if (ai →exp == -1) return c;
10             int sum = ai →coef + bi →coef;
11             if (sum) c.poly.InsertBack (temp.Set (sum, ai →exp));
12             ai ++; bi ++; // advance to next term
13          }
14          else if (ai →exp < bi →exp) {
15                c.poly.InsertBack (temp.Set (bi →coef, bi →exp));
16                bi ++; // next term of b
17          }
18          else {
19                c.poly.InsertBack (temp.Set(ai →coef , ai →exp));
20                ai ++; // next term of a
21          }
22       }
23 }
```

- مزیت نسبت به زنجیر:
  - عدم نیاز به دو حلقه آخر

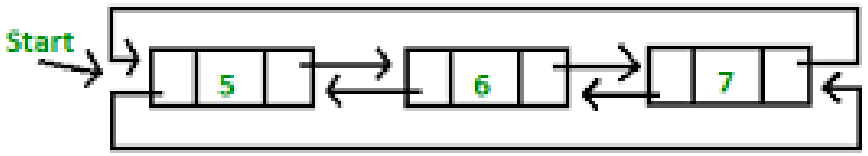
لیست های دو پیوندی

- زنجیر
- لیست حلقوی یک پیوندی
- مشکلات: فقط می توانیم در جهت پیوندها حرکت کنیم ⟵ تنها راه یافتن گره ماقبل p پیمایش از ابتدای لیست است.
- ایجاد مشکل در حذف یک گره دلخواه

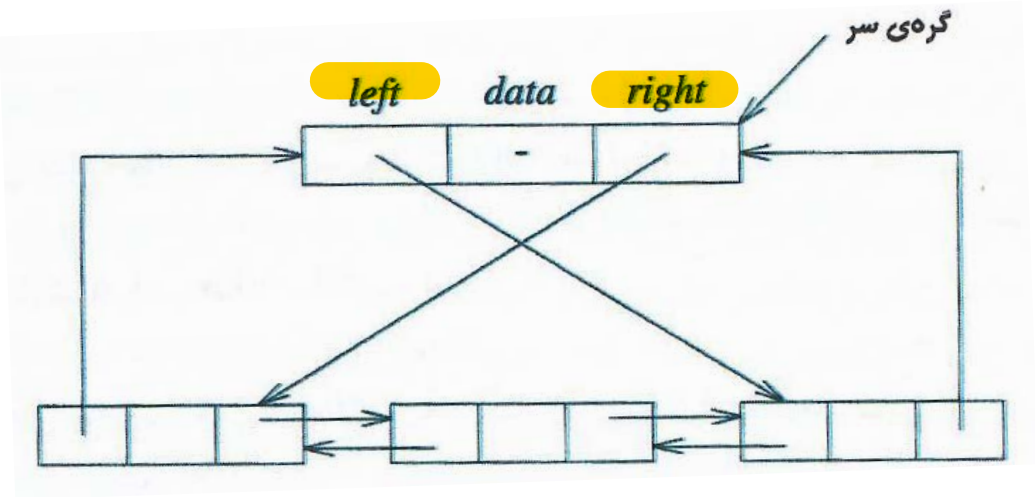
لیست های دو پیوندی

- لیست دو پیوندی ساده

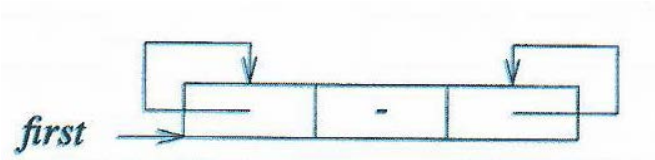
- لیست دو پیوندی حلقوی



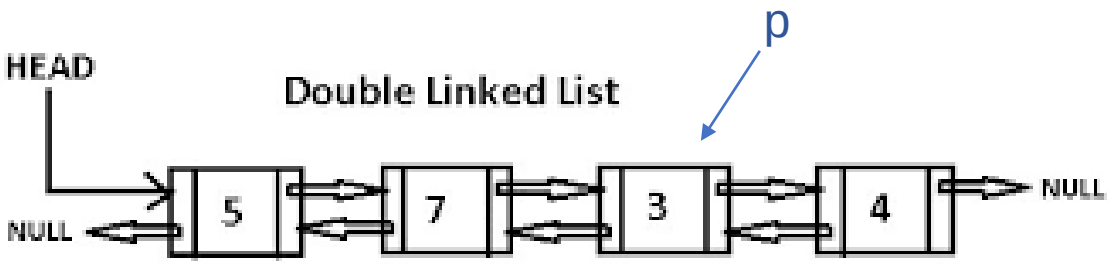
لیست دو پیوندی با گره سر



لیست حلقوی دو پیوندی خالی با گره سر



مزیت لیست های دو پیوندی



$$p = p \rightarrow left \rightarrow right = p \rightarrow right \rightarrow left$$

تعریف کلاس یک لیست دو پیوندی

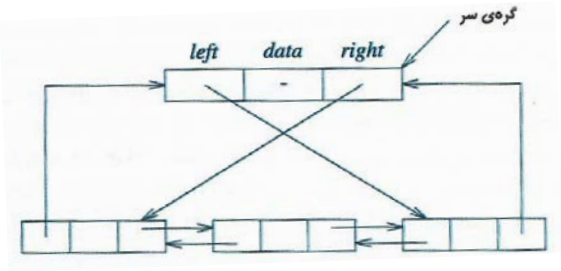
```
class DblList;

class DblListNode {
friend class DblList;
private:
    int data;
    DblListNode *left, *right;
};

class DblList {
public:
    // List manipulation operations
    .
    .
private:
    DblListNode *first; // points to header node
};
```

حذف یک گره از یک لیست حلقوی دو پیوندی

```
void DblList::Delete(DblListNode *x)
{
   if (x == first) throw "Deletion of header node not permitted";
   else {
      x → left → right = x→right;
      x → right → left = x →left;
      delete x;
   }
}
```

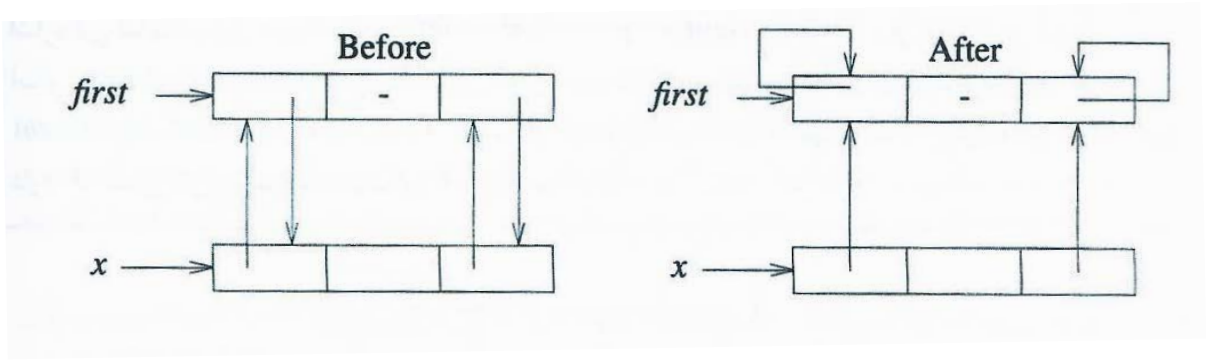


سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

حذف یک گره از یک لیست حلقوی دو پیوندی

- حذف از لیستی که تنها یک نود غیر از head دارد.

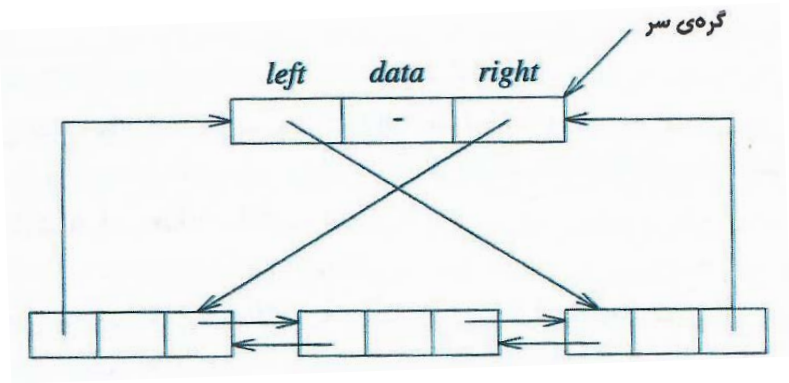
```
void DblList::Delete(DblListNode *x)
{
   if (x == first) throw "Deletion of header node not permitted";
   else {
      x → left → right = x→right;
      x → right → left = x →left;
      delete x;
   }
}
```



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

اضافه کردن یک گره به یک لیست حلقوی دو پیوندی

```
void DblList::Insert(DblListNode *p, DblListNode *x)
{// insert node p to the right of node x
    p →left = x; p →right = x →right;
    x →right →left = p; x →right = p;
}
```



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