لیست پیوندی یا linked list:

در لیست پیوندی هر عنصر آدرس عنصر بعدی را در خود نگه می دارد نحوه دسترسی آن به این صورت است که اگر بخواهیم به عنصر سوم برسیم ابتدا به عنصر اول و سپس به عنصر دوم و بعد به عنصر سوم خواهیم رسید. لیست پیوندی ساختمانی است که یکی از اجزایش اشاره گری از همان ساختمان است. به عنصر اول لیست، سر لیست یا head و به عنصر آخر لیست tail می گویند. اشاره گر عنصر آخر لیست به null اشاره می کند.

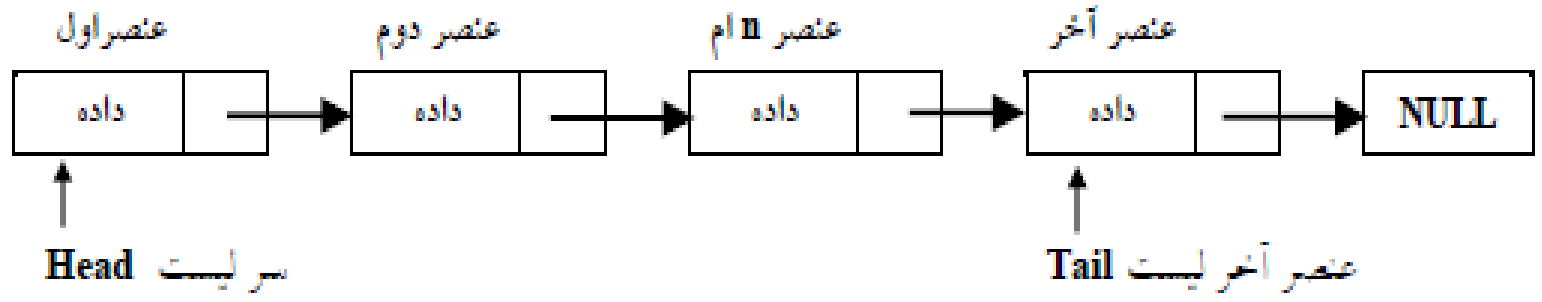
لیست پیوندی

آدرس شروع لیست را درون یک اشاره گر به نام head می ریزیم و آن را دستکاری نمی کنیم.

head = آدرس اولین عنصر

آدرس دومین عنصر= head→ next

آدرس سومین عنصر = عنصر دوم → next



لیست پیوندی

- اشاره گر سر و آخر لیست معمولا به صورت سراسری تعریف می شوند

Std *Head=null , *Tail=null

عملیات لیست پیوندی

- ایجاد لیست
- درج گره در لیست
- حذف گره از لیست
- جستجو در لیست
- مرتب سازی لیست
- معکوس کردن لیست
- و ...

لیست پیوندی

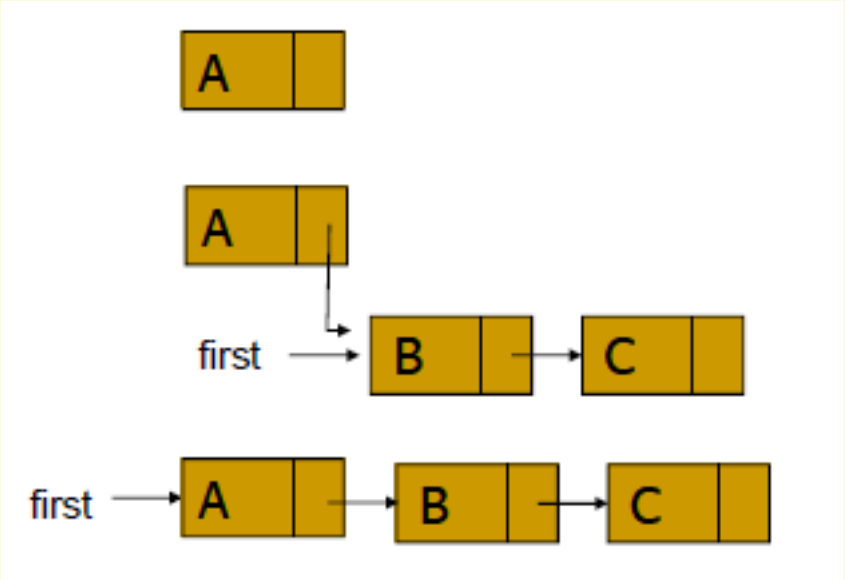
```
class CNode
 {
private:
char data;
CNode *link;
public:
CNode(char d , Cnode *l=NULL);
};
```

کلاس لیست پیوندی

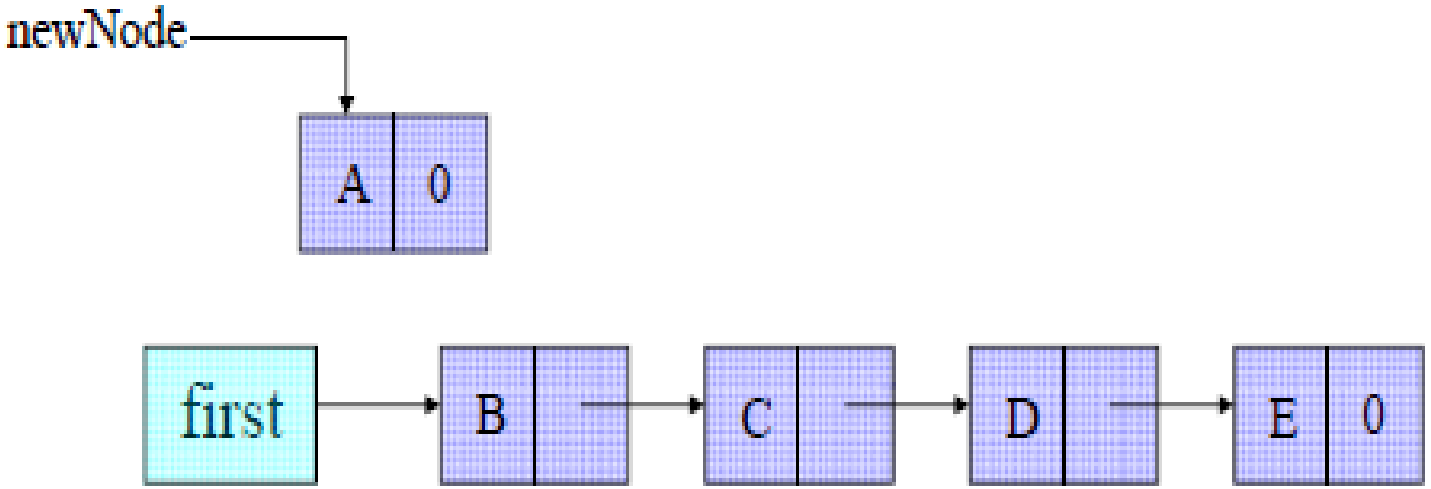
```
class CLinkList
{
private:
CNode *first,*last;
public:
CLinkList();
CLinkList(const CLinkList &l);
void InsertFirst(char new_char);
void InsertAfter(CNode *pNode , char new_char);
void InsertBefore(CNode *pNode , char new_char);
void InsertLast(char new_char);
void DeleteItem(CNode *pNode);
char GetItemData(int ItemIndex);
void Display()const;
CNode* Find(char ch);
void DestroyList();
~CLinkList();
};
```

درج گره در ابتدای لیست

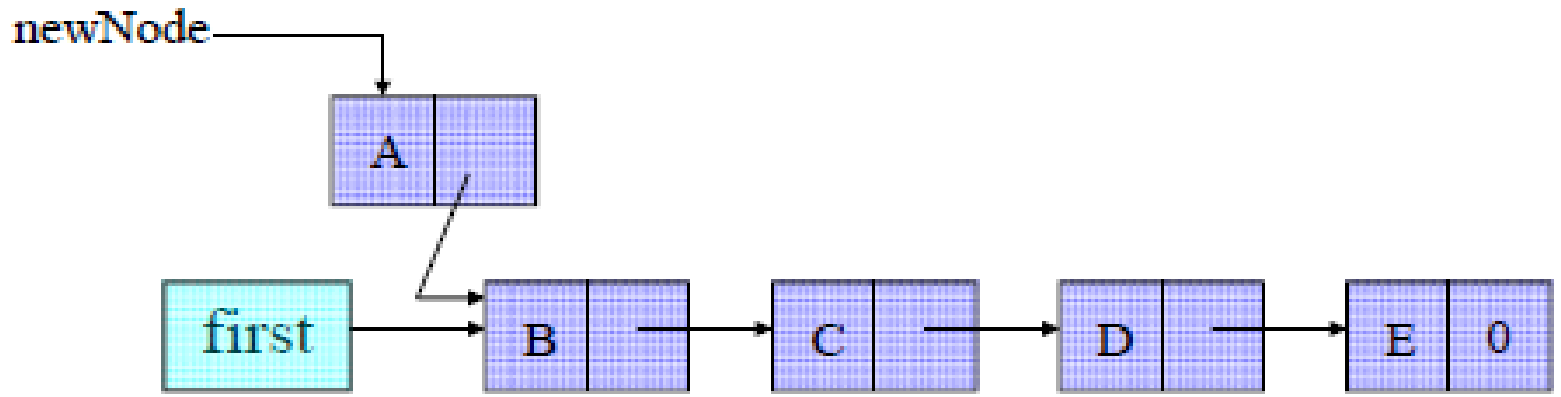
```
void CLinkList::InsertFirst (char new_char)
{     CNode *t = new CNode(new_char, first);
first = t;
if(!last)
last = first;
}
```



اضافه کردن به ابتدای لیست

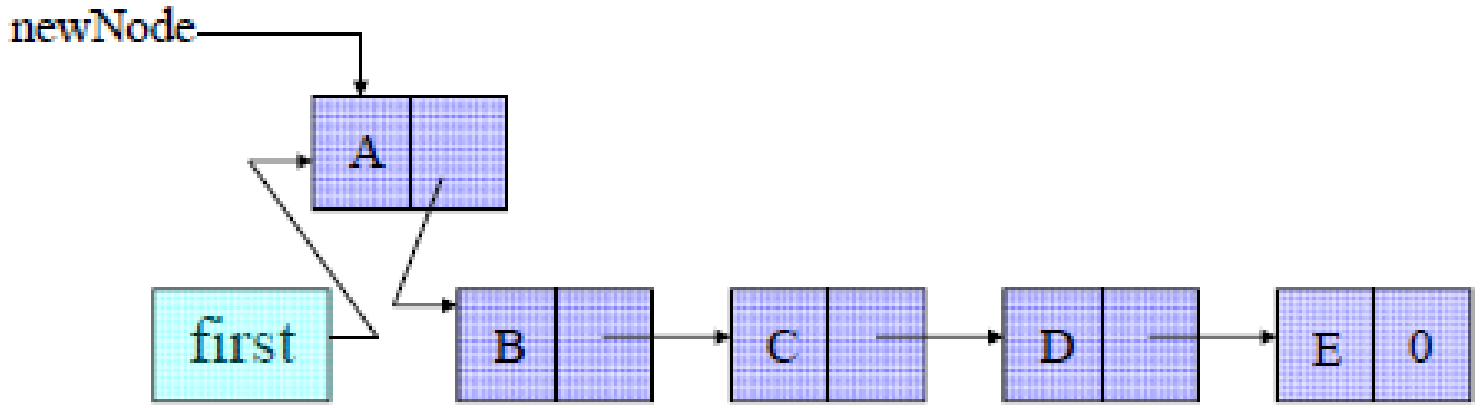


اضافه کردن به ابتدای لیست-ادامه



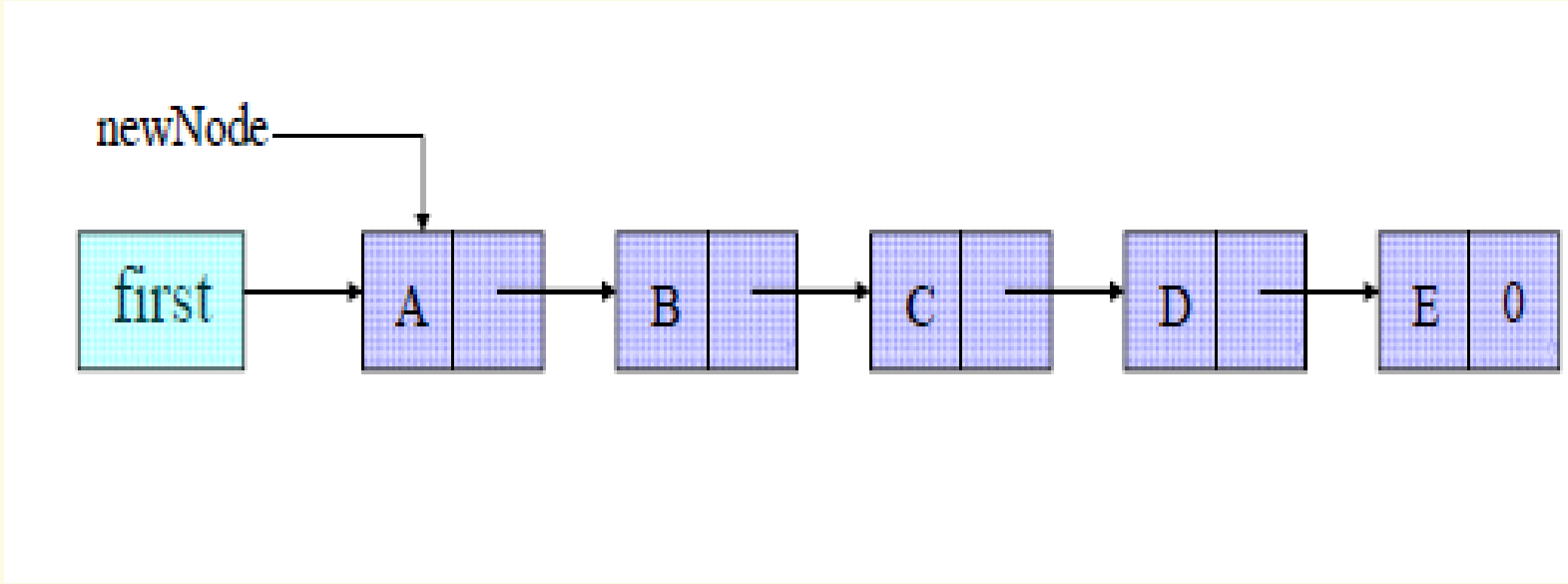
newNode → link = first;

اضافه کردن به ابتدای لیست




```
first = newNode;
```

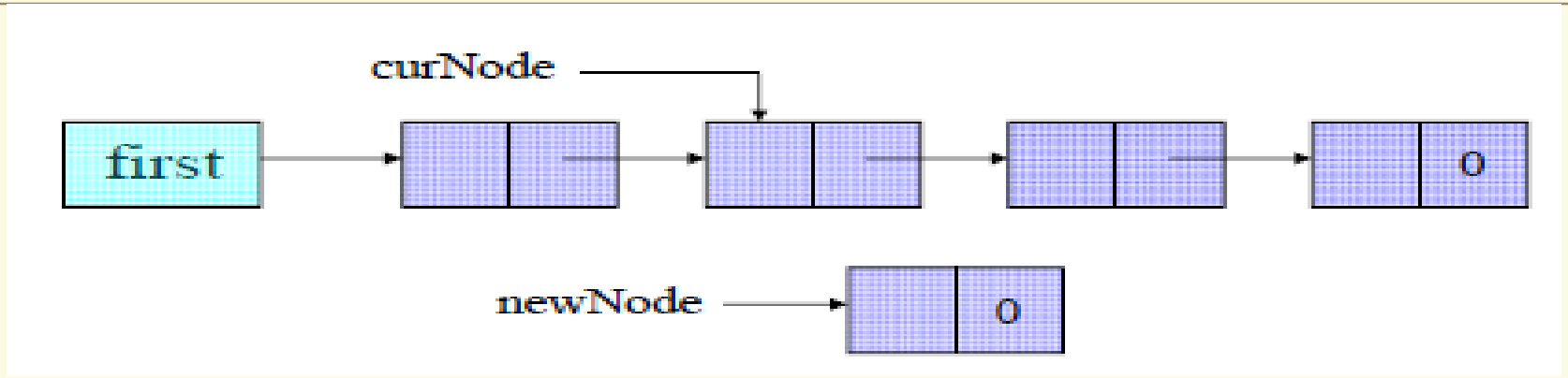
اضافه کردن به ابتدای لیست

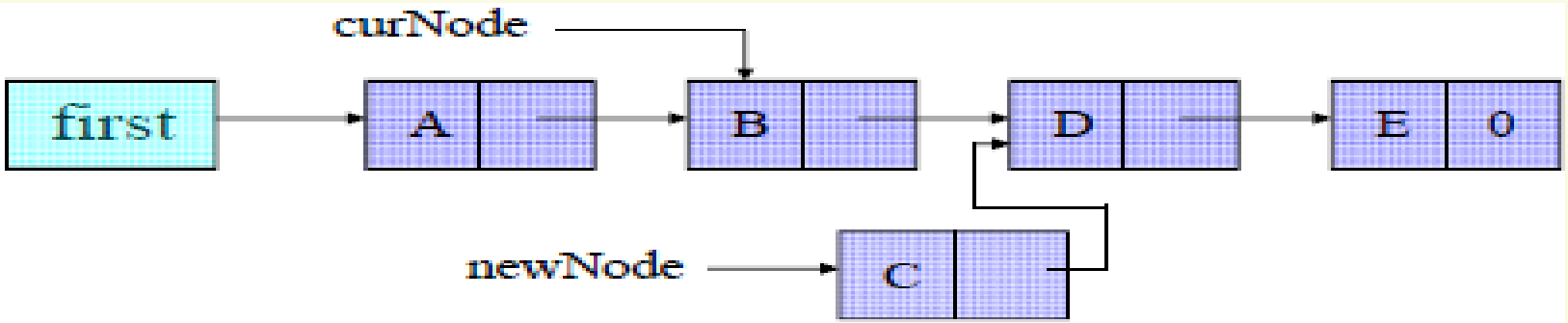


نمایش لیست پیوندی (پیمایش)

```
void CLinkList ::Display()
{
if( !first ) return;      //no node
Node *temp = first;
while(temp)               //traverse list
{
cout << temp→data;
temp = temp→link; //next node
}
}
```

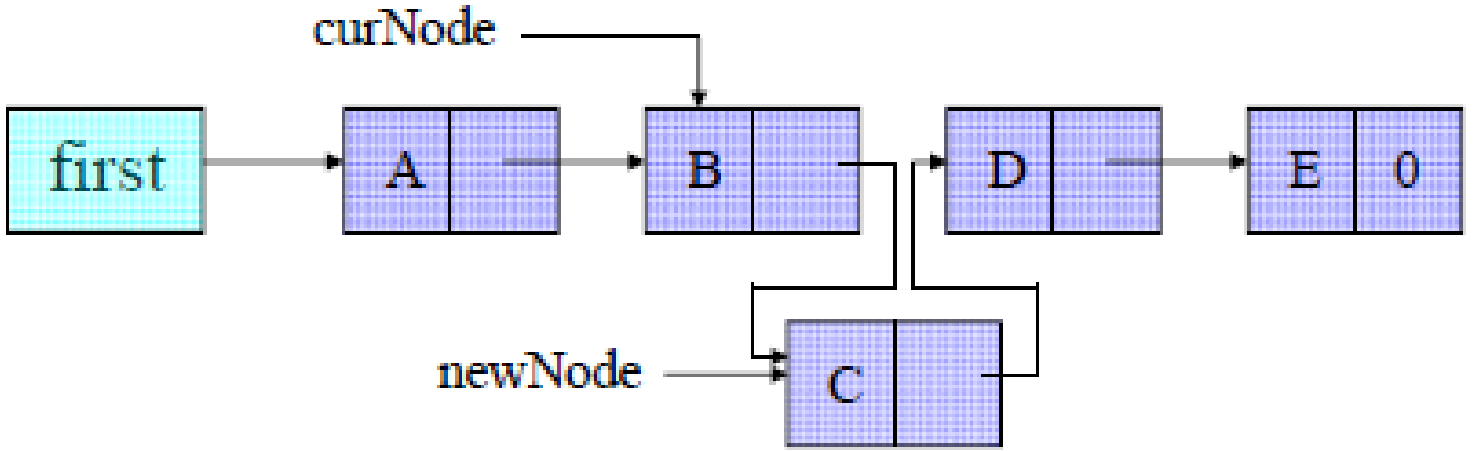
درج عنصر جدید





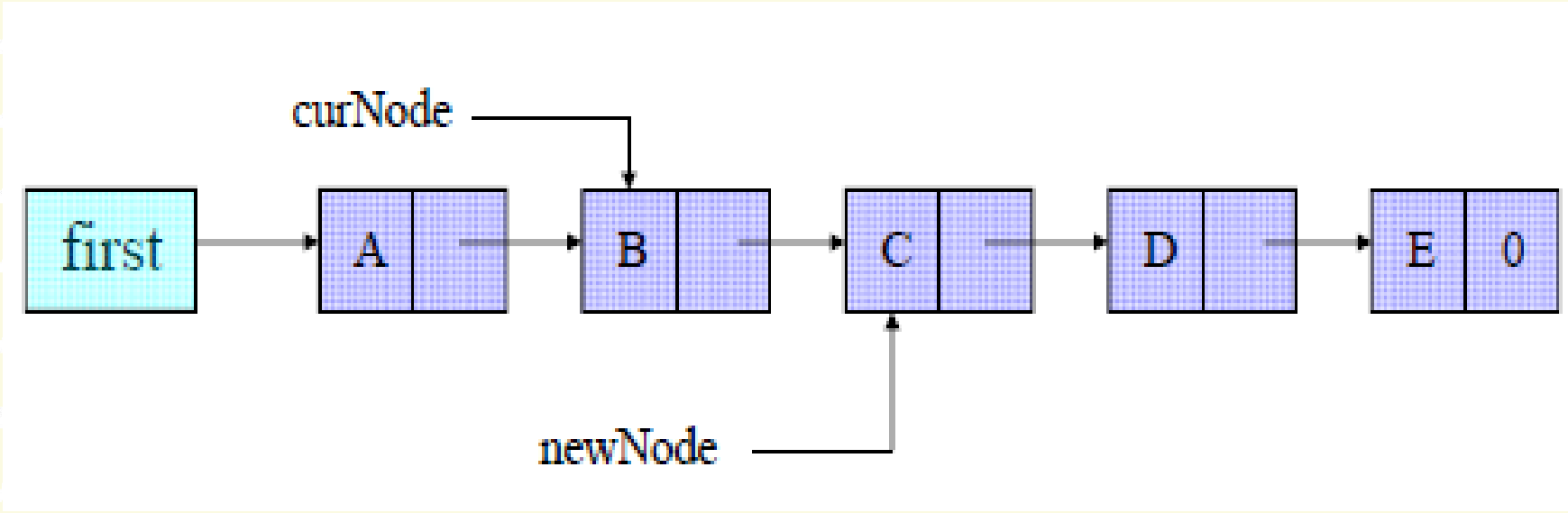

```
newNode → link = curNode → link ;
```

اضافه کردن عنصر جدید



curNode → link = newNode ;

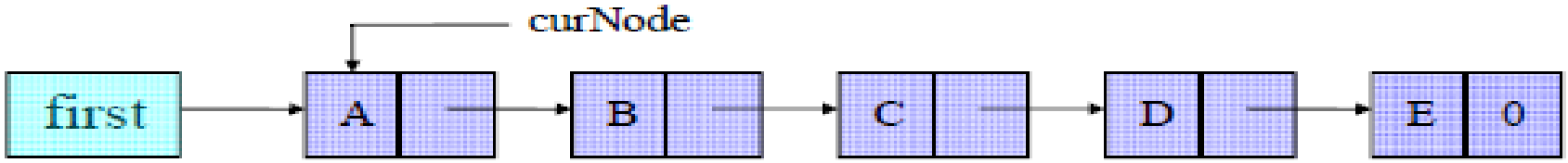
اضافه کردن عنصر جدید

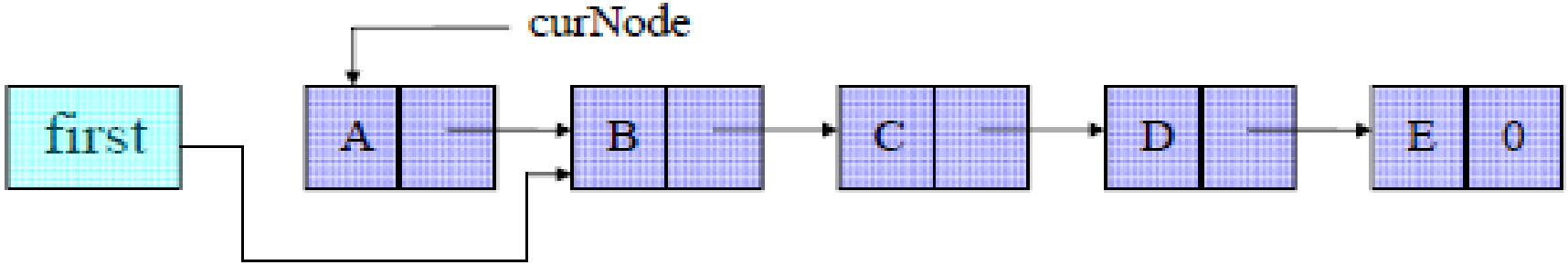


اضافه کردن عنصر جدید

```
void CLinkList::InsertAfter (CNode *pNode , char new_char)
{
CNode*newNode = new CNode(new_char);
newNode→link = curNode→link ; //assign newNode
curNode→link = newNode ; //assign curNode
}
```

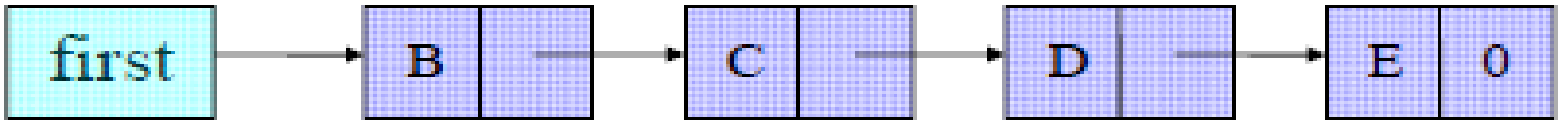
حذف عنصر از ابتدای لیست پیوندی





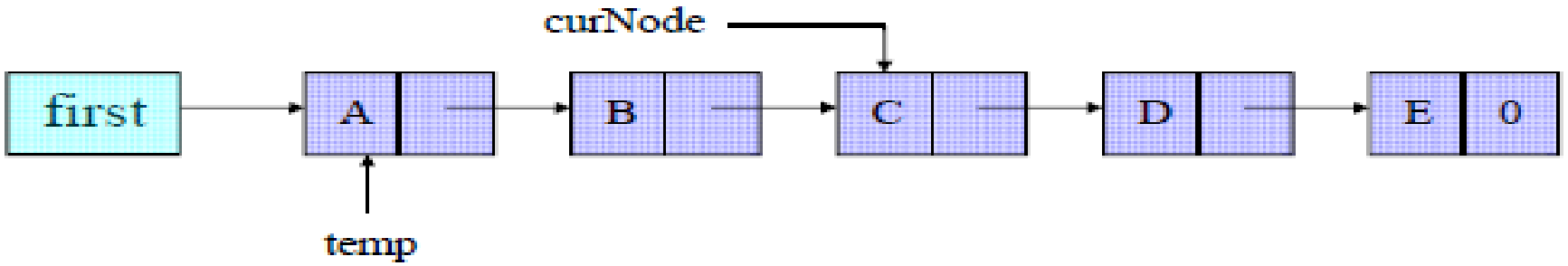
first = first →link;

حذف عنصر از ابتدای لیست پیوندی



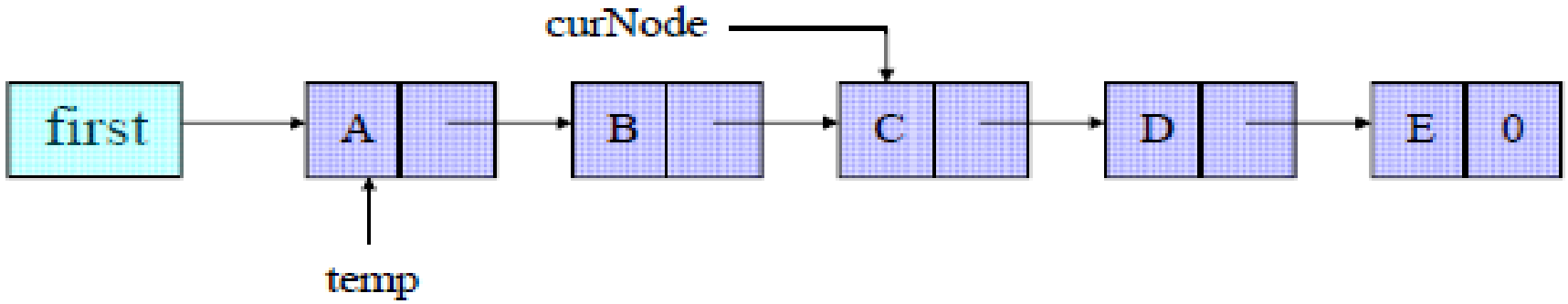
delete cur;

حذف عنصر از لیست پیوندی



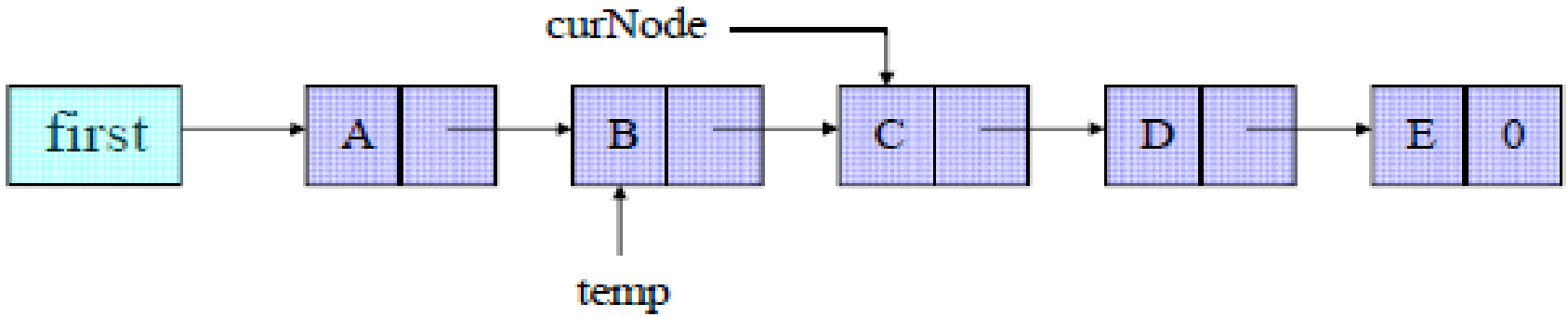

```
Node *temp = first;
```

حذف عنصر از لیست پیوندی



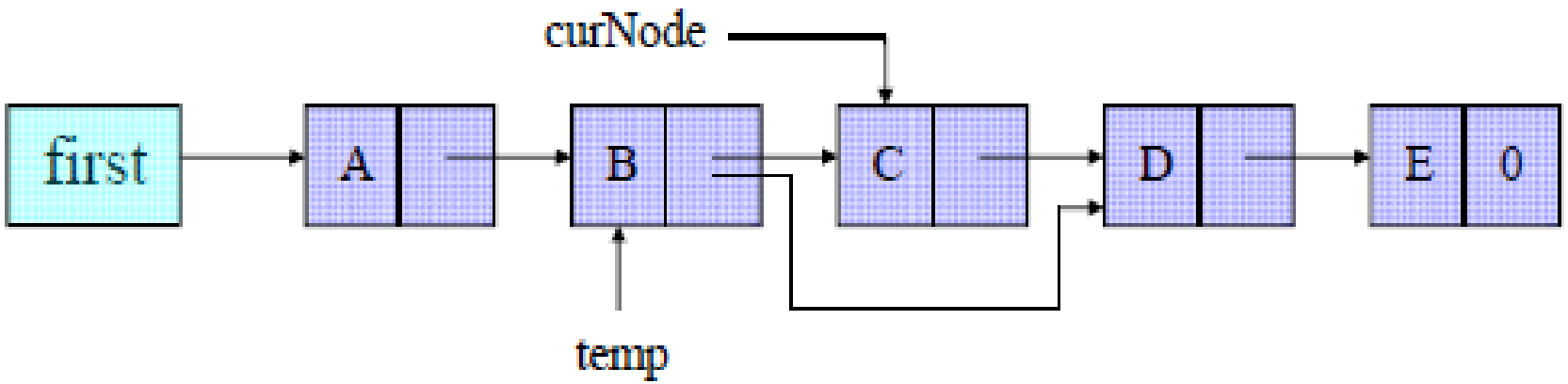

```
while( temp → link != cur)
        temp = temp → link;
```

حذف عنصر از لیست پیوندی



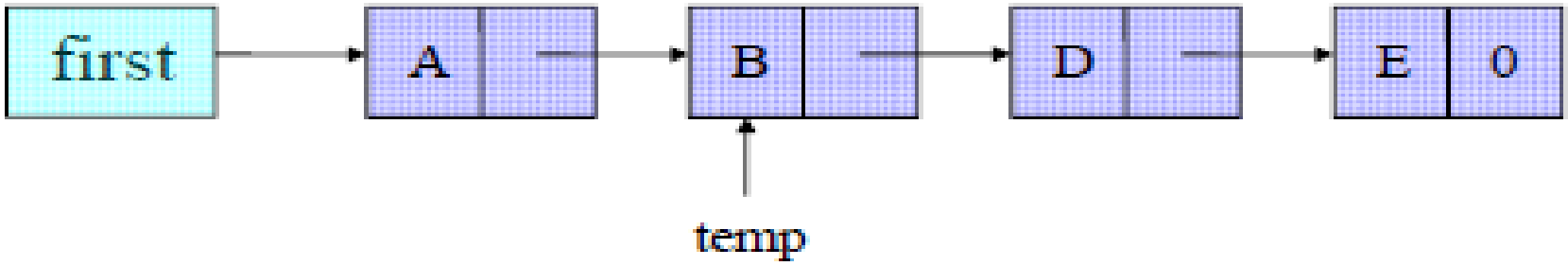

```
while( temp → link != cur)
        temp = temp → link;
```

حذف عنصر از لیست پیوندی



temp → link = cur → link;

حذف عنصر از لیست پیوندی



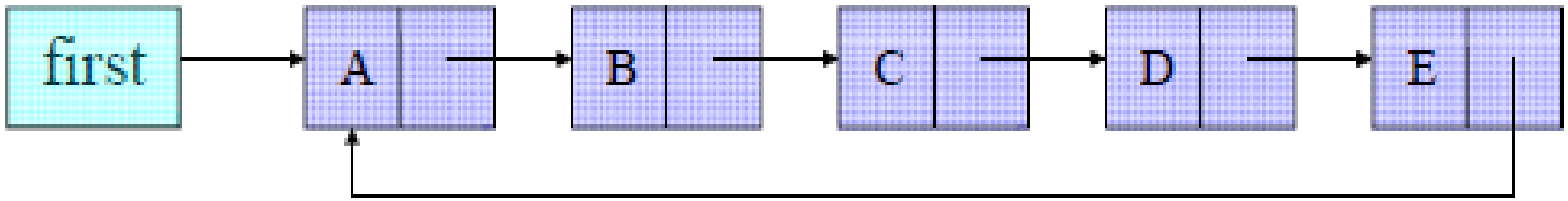

```
delete cur;
```

حذف عنصر از لیست پیوندی

```
void CLinkList::DeleteItem(CNode *cur)
{
if( cur == first )
{
first = first → link;
delete cur;
return;
}
Node *temp = first;
while( temp→link != cur)
temp = temp → link;
Temp→link = cur→link;
delete cur;
}
```

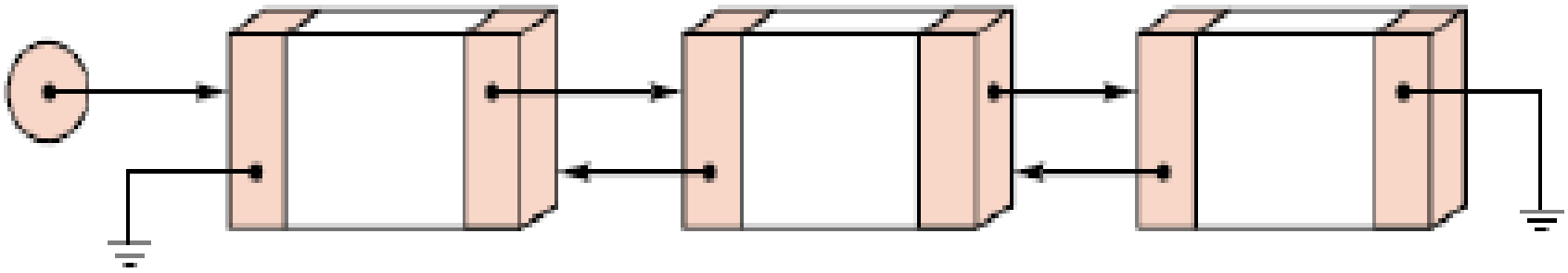
لیست های حلقوی

- به وسیله ی اتصال آخرین گره به اولین گره می توان لیست حلقوی را تولید کرد.
- در این گونه لیست ها به راحتی می توان از آخر لیست به اول لیست پرش کرد.



لیست های پیوندی دوگانه (دو طرفه)

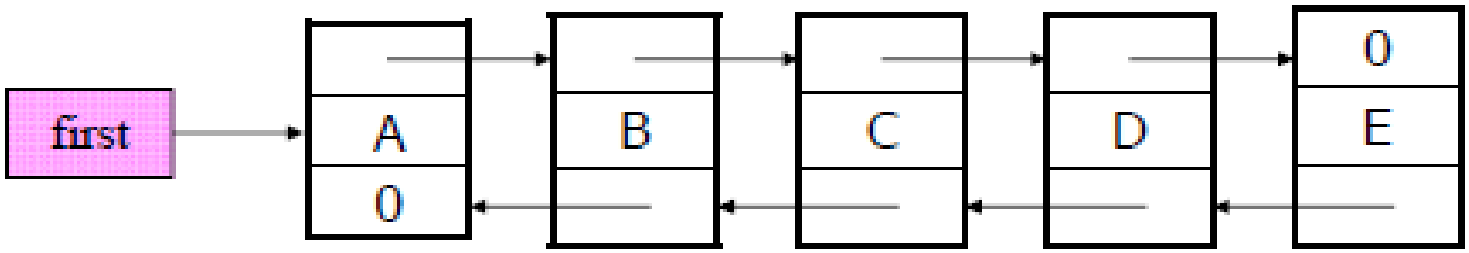
- مشکل اساسی لیست های یک طرفه عدم دسترسی به گره قبلی
- لیست های دوگانه با اضافه کردن یک اشاره گر به گره ی قبلی این مشکل را حل کرده است.



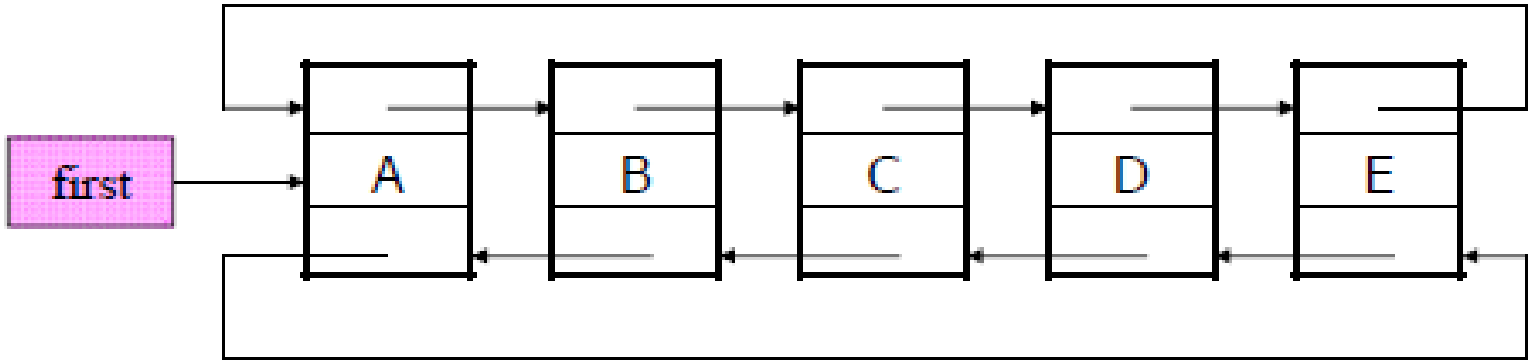
A doubly linked list

انواع لیست های پیوندی دوگانه

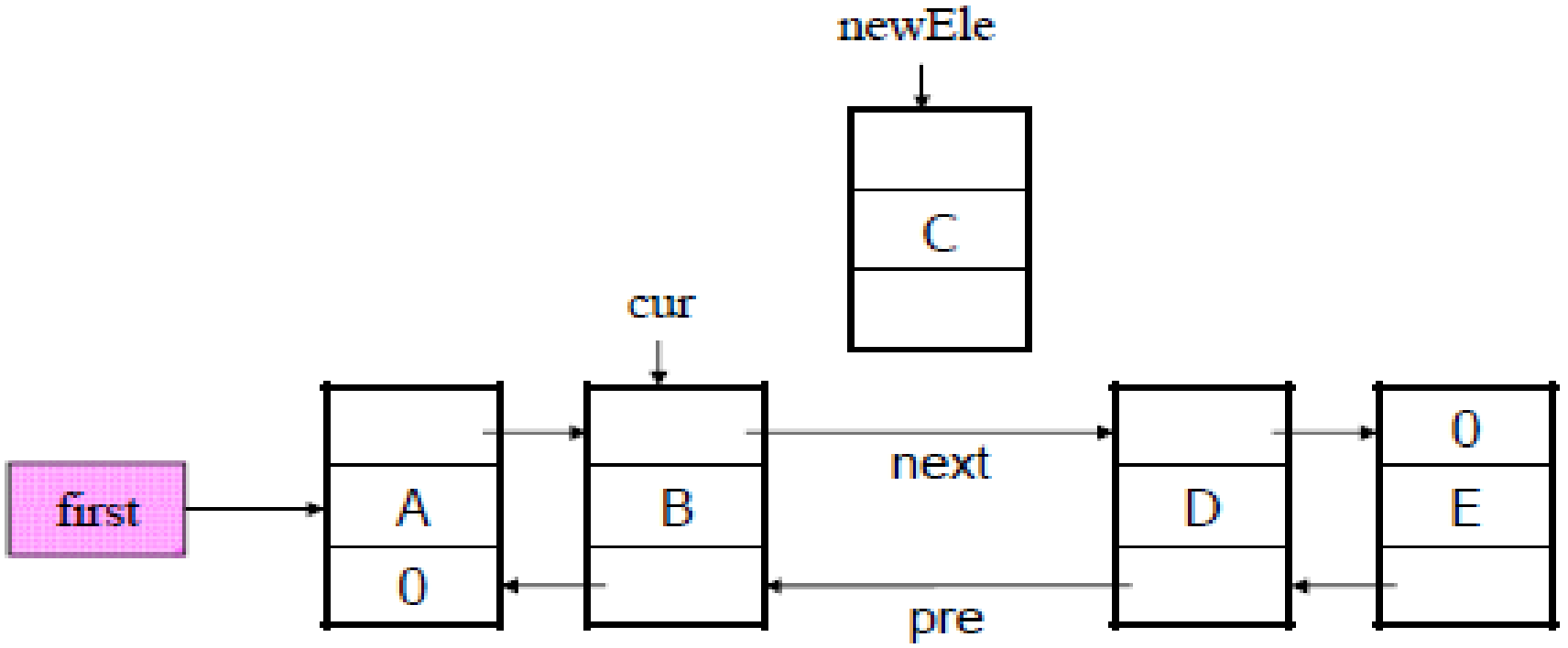
لیست دوطرفه ی ساده



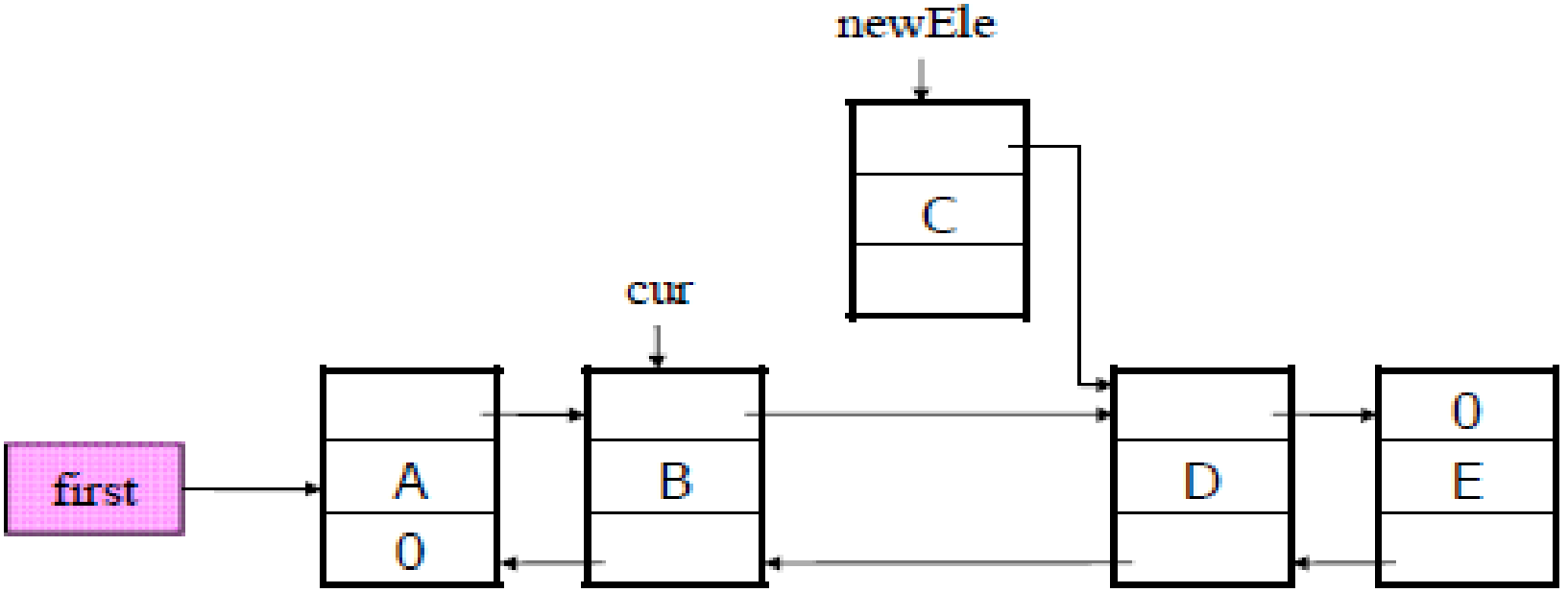
لیست دوطرفه ی حلقوی



اضافه کردن عنصر به لیست دوطرفه ساده

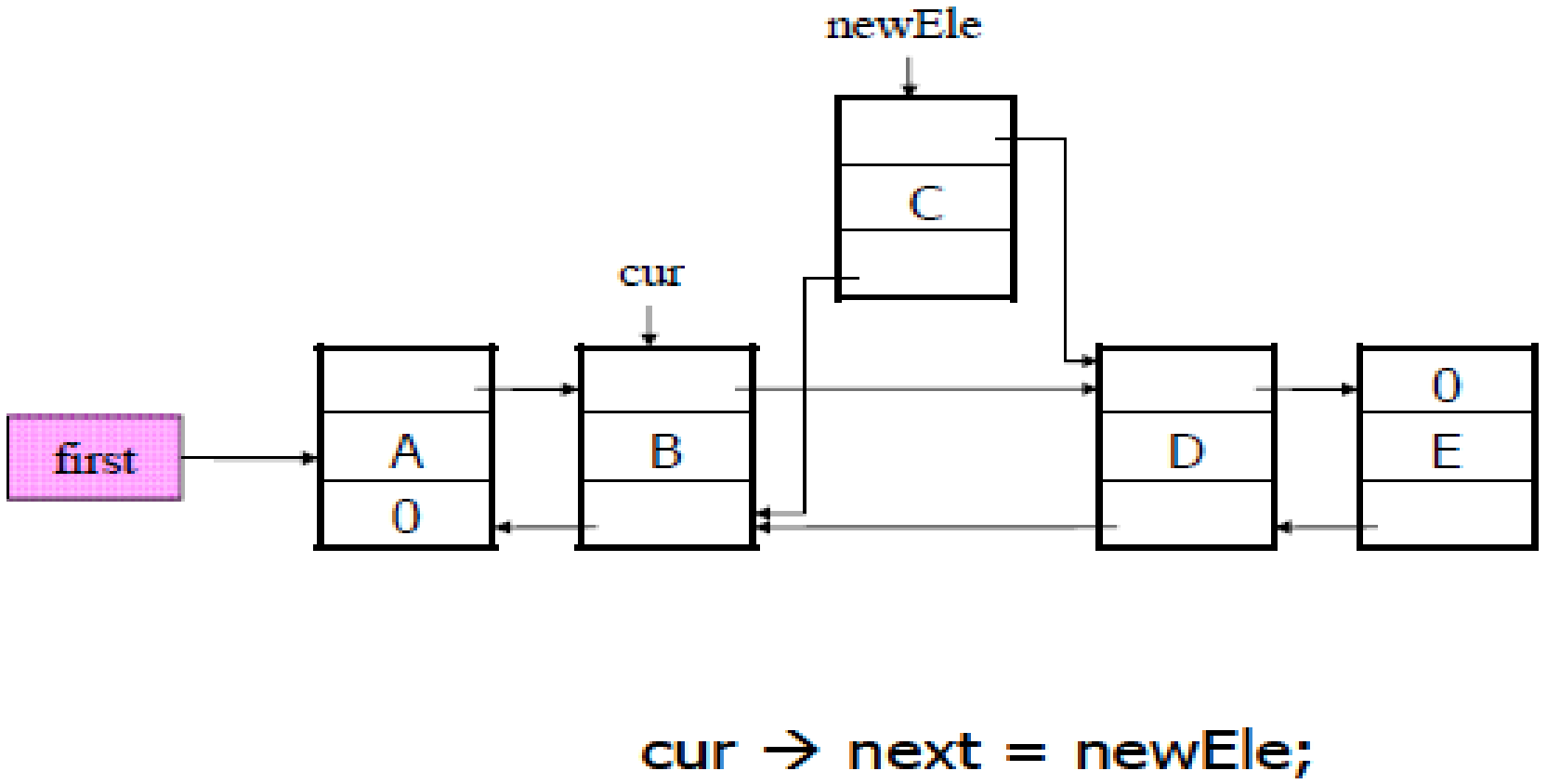


اضافه کردن عنصر به لیست دوطرفه ساده

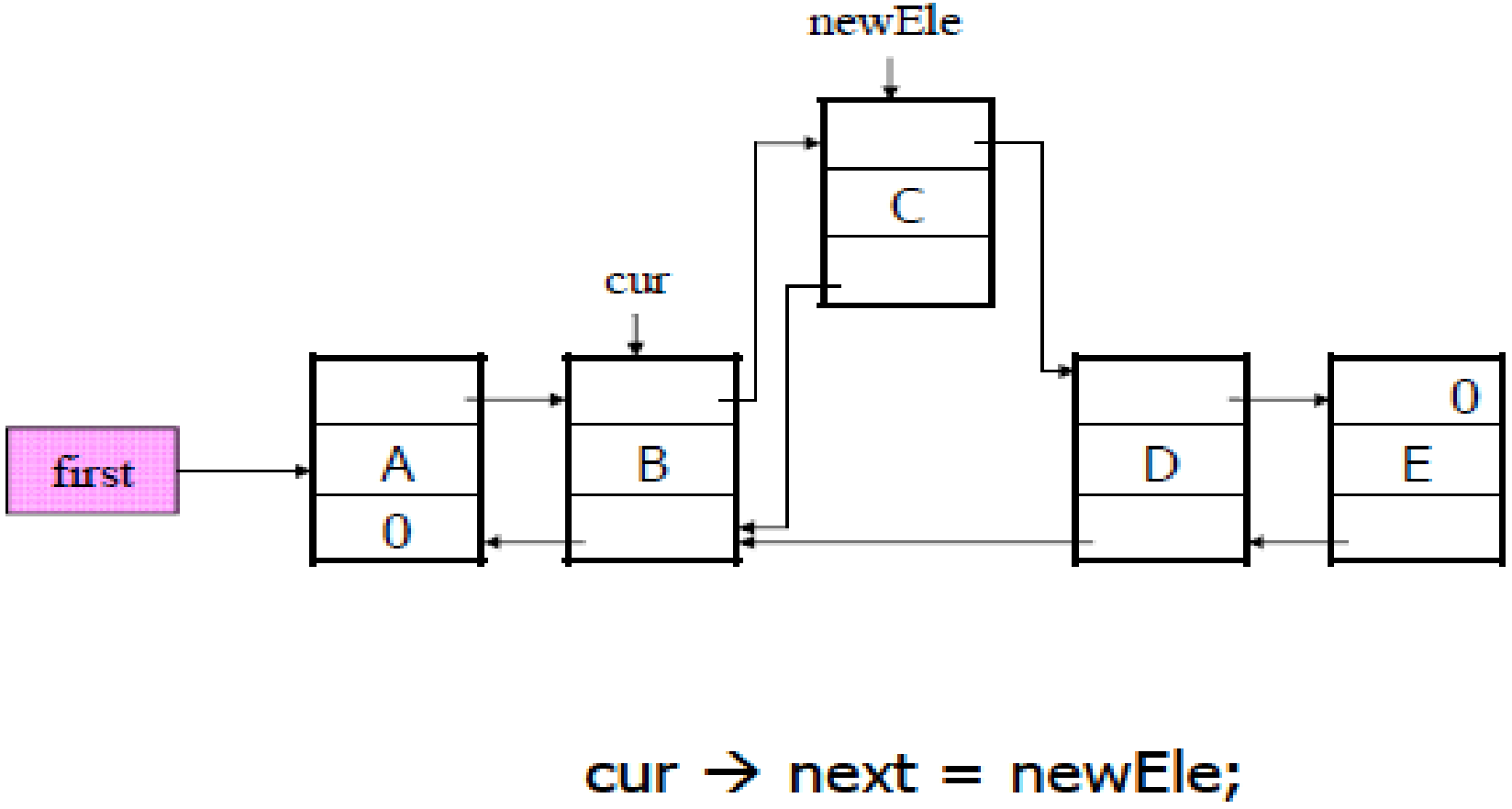


newEle → next = cur → next;

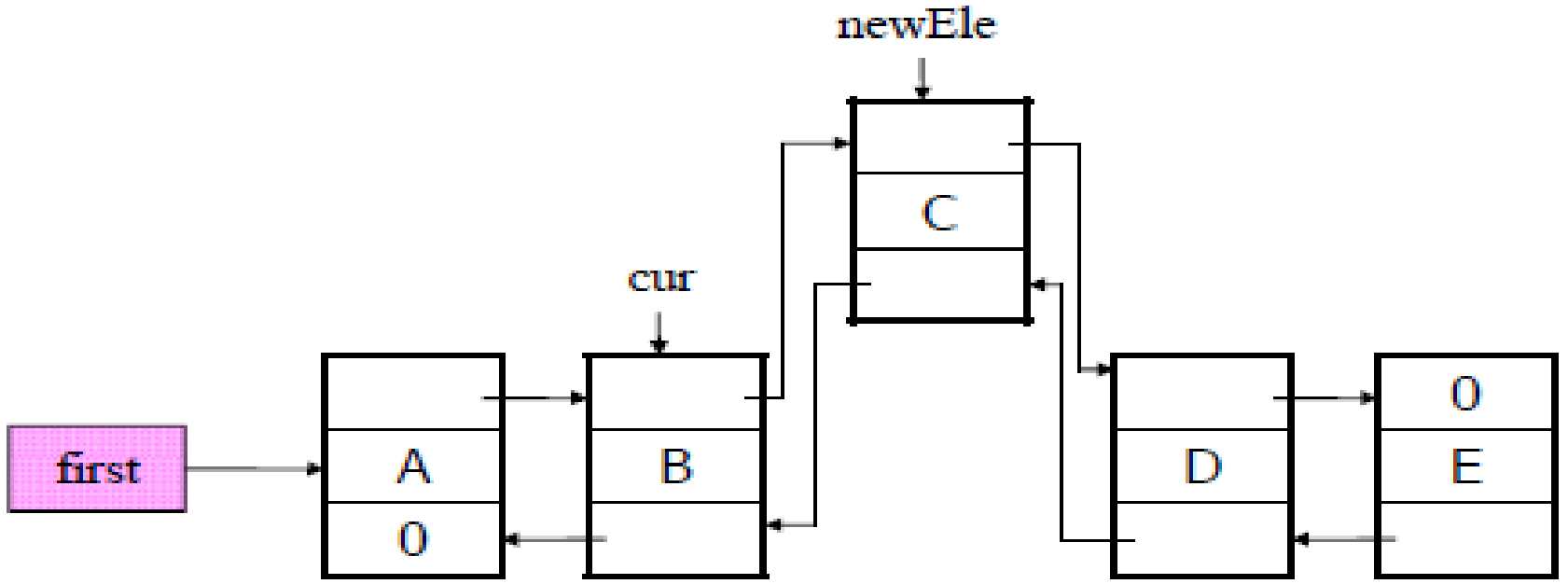
اضافه کردن عنصر به لیست دوطرفه ساده



اضافه کردن عنصر به لیست دوطرفه ساده

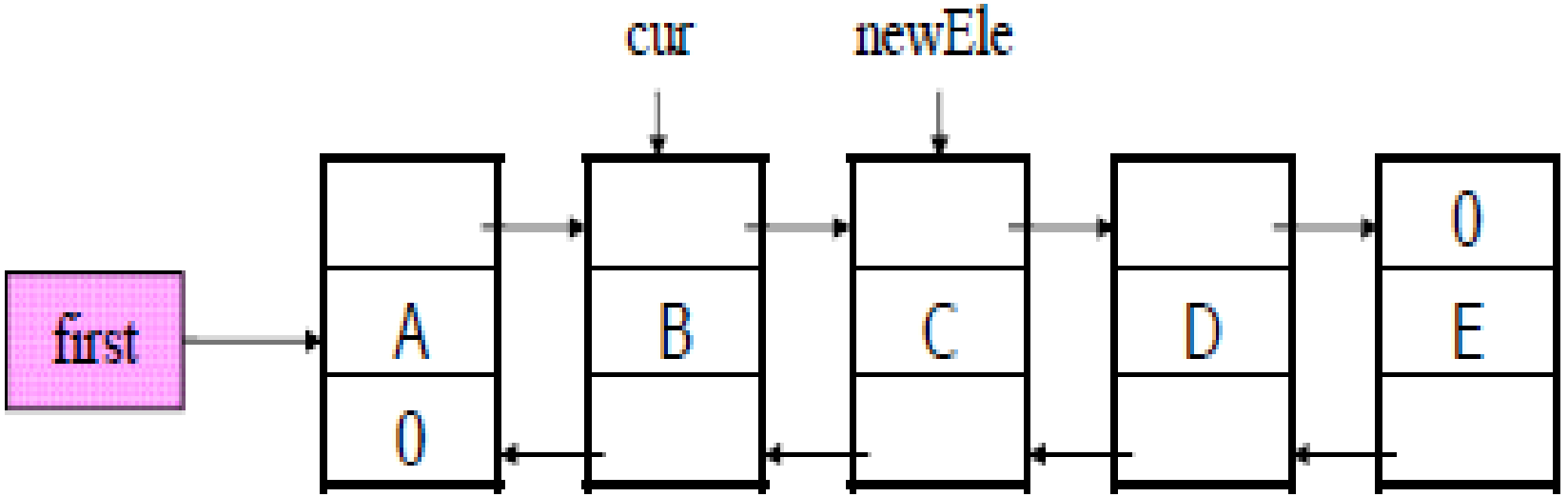


اضافه کردن عنصر به لیست دوطرفه ساده

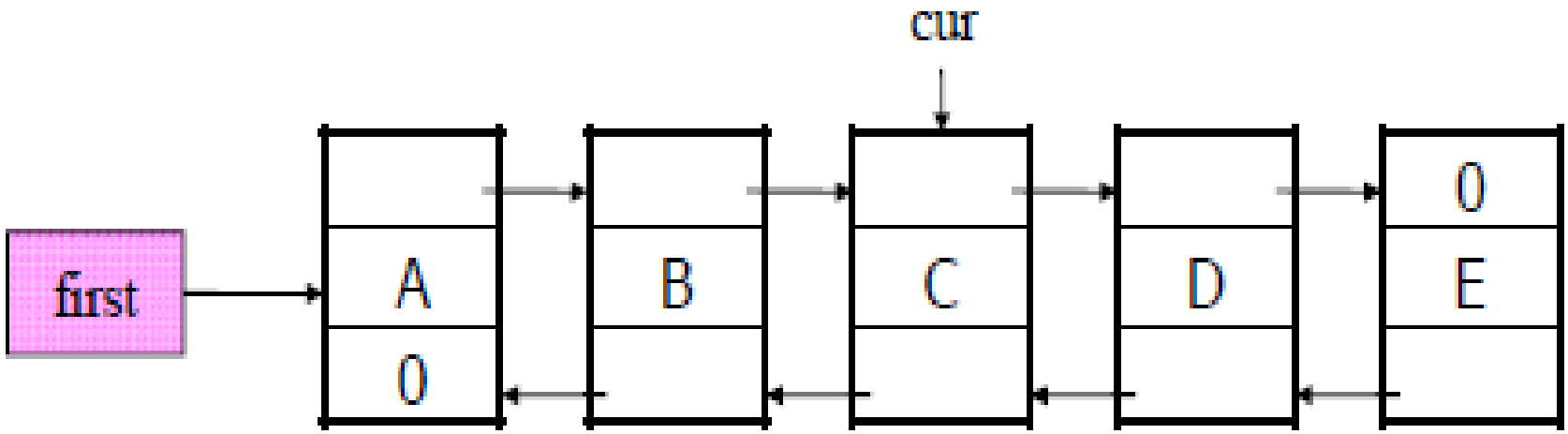


(newEle → next) → pre = newEle;

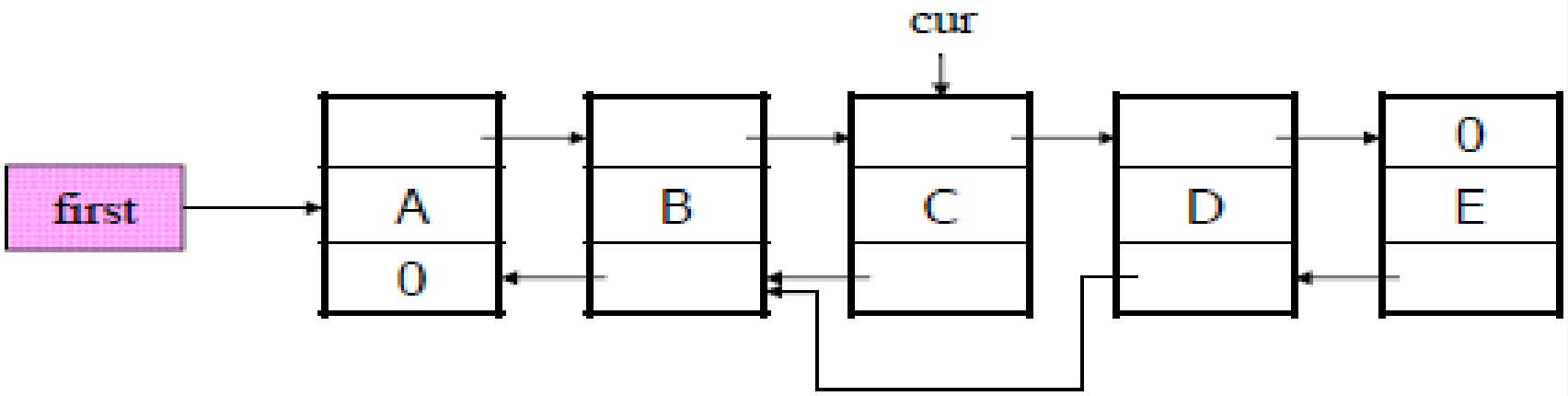
اضافه کردن عنصر به لیست دوطرفه ساده



حذف از لیست دوطرفه ساده

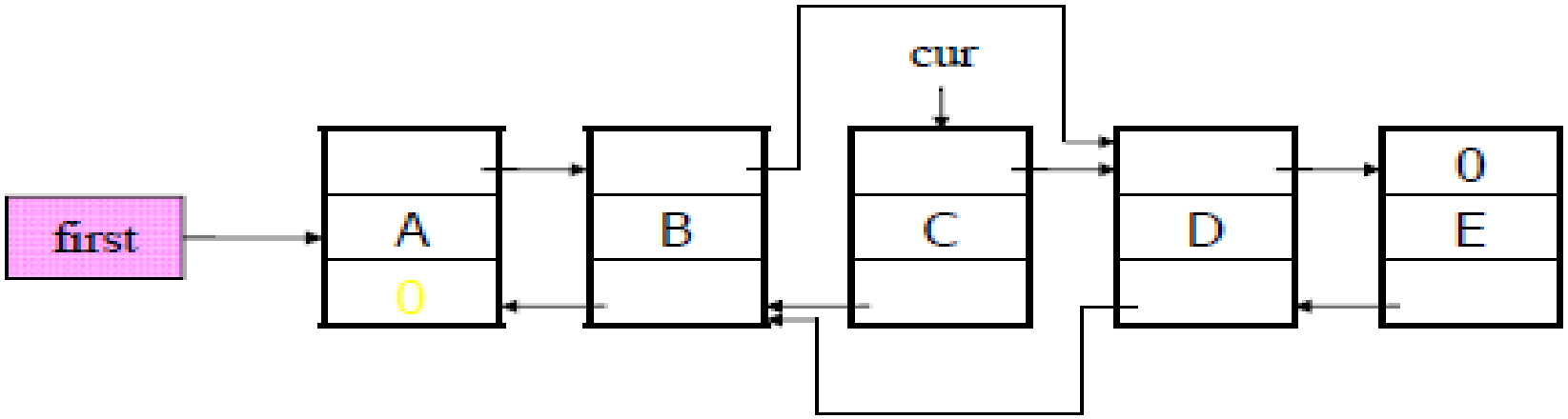


حذف از لیست دوطرفه ساده



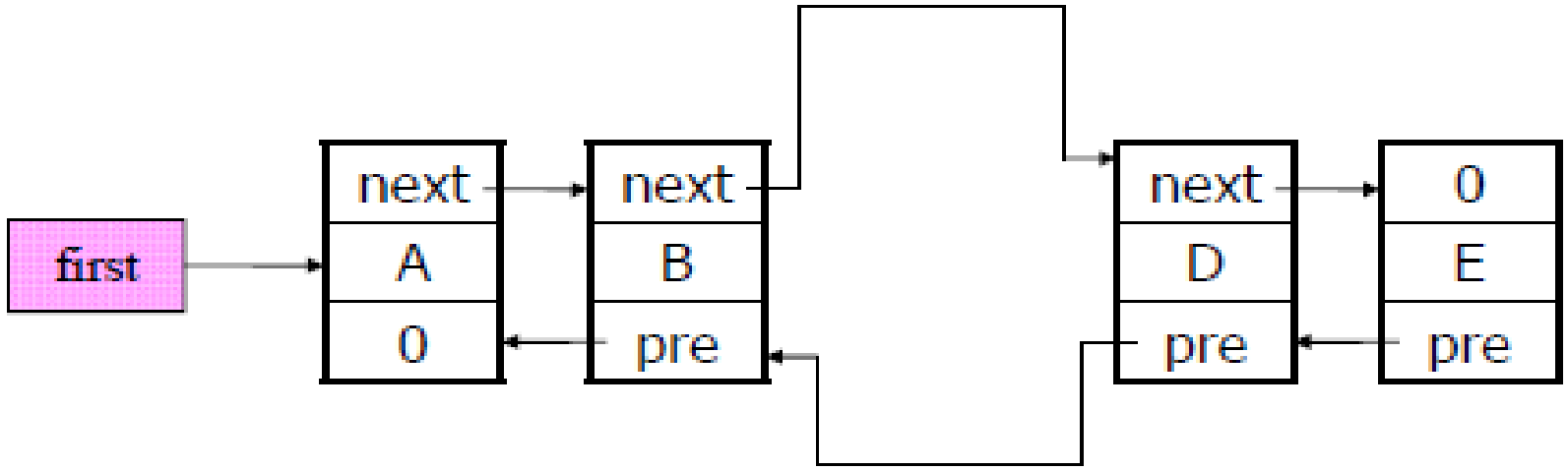
(cur → next) → pre = cur → pre;

حذف از لیست دوطرفه ساده



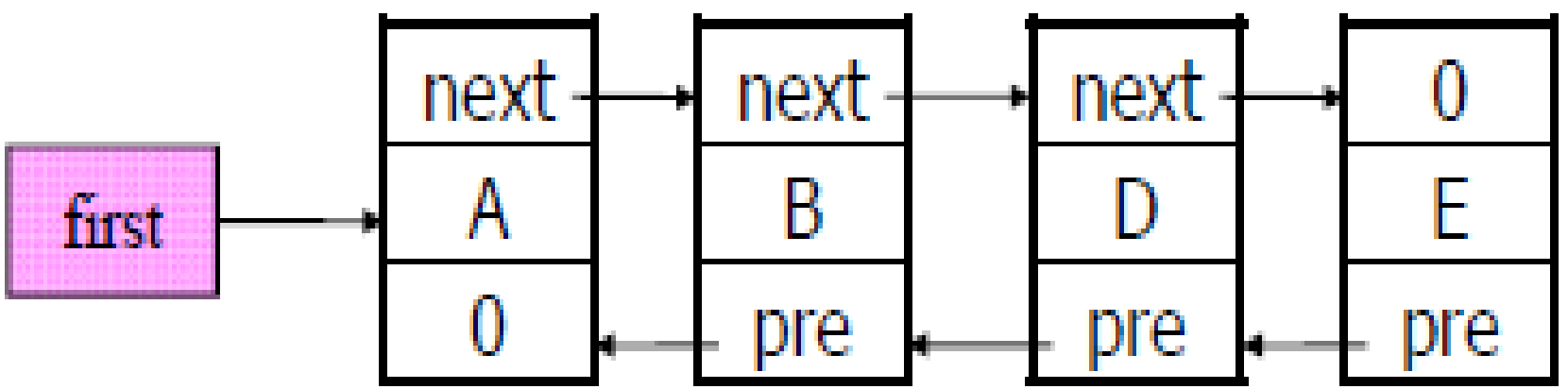
(cur → pre) → next = cur → next;

حذف از لیست دوطرفه ساده



(cur → pre) → next = cur → next;

حذف از لیست دوطرفه ساده



جستجو کردن در لیست:

تابعی بنویسید که فامیلی را به عنوان ورودی گرفته و درون لیست جستجو می کند و آدرس عنصری که فامیل آن با فامیل رکورد ورودی مساوی باشد را برگرداند.

```
#include <iostream>
#include <conio.h>
std * find(char * family)
 {
  std * temp = head;
  while (temp != null)
  {
   if(strcmp(family,temp->family)==0)
    break;
   temp = temp-> next;
  }
  return temp;
 }
```