ساختمان داده‌ها و الگوریتم‌ها

فصل سوم

صمد نجارقابل

S.Najjar.G@Gmail.com

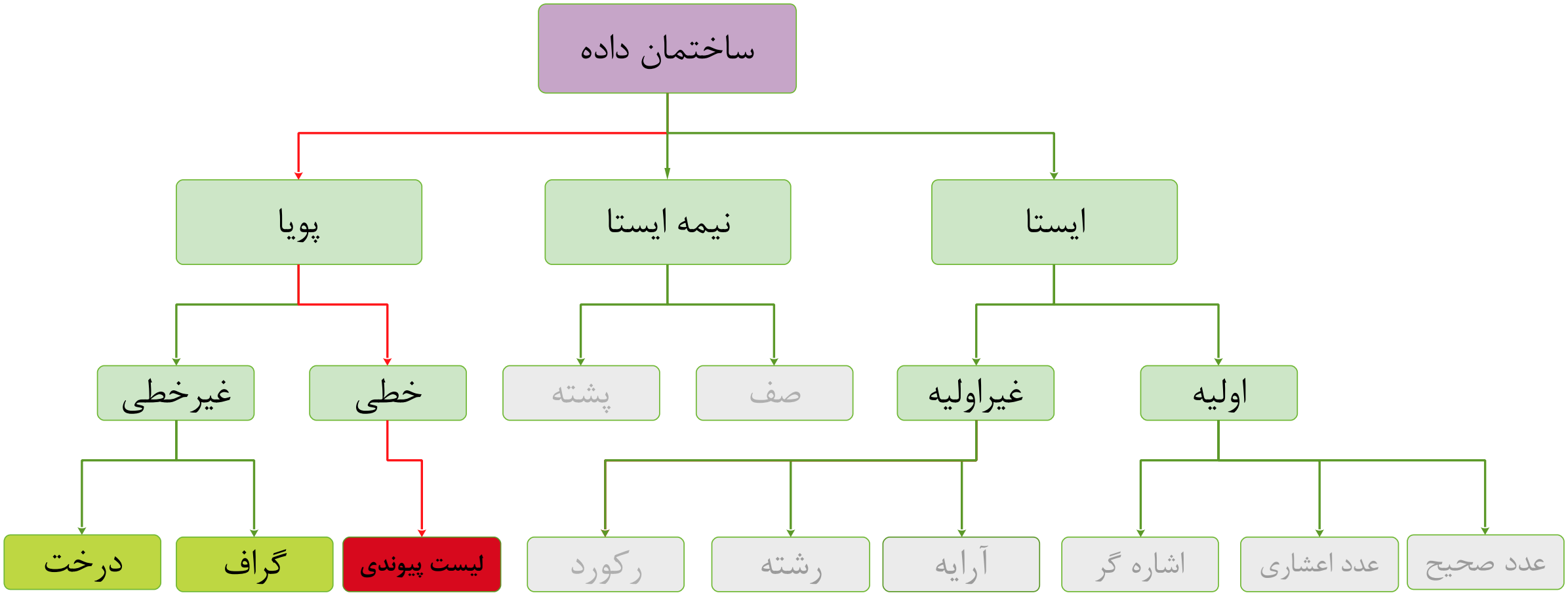
S.Najjar.G@Gmail.com

فهرست مطالب

- مقدمه‌ای بر الگوریتم‌ها و مفاهیم پایه
- معرفی پیچیدگی زمانی و حافظه‌ای و روشهای تحلیل مسائل
- **معرفی ساختمان داده‌های مقدماتی و الگوریتم‌های وابسته به آنها**
  - آرایه
  - صف
  - پشته
  - **لیست پیوندی**
- تئوری درخت و گراف و الگوریتم‌های مرتبط
- الگوریتم‌های مرتب‌سازی و تحلیل پیچیدگی مربوط به آنها
- مباحث تکمیلی در ساختمان داده‌ها

S.Najjar.G@Gmail.com

دسته بندی ساختمان داده‌ها



S.Najjar.G@Gmail.com

مقدمه

- لیست مرتب زیر را در نظر بگیرید

(*bat, cat, sat, vat*)

  - برای اضافه کردن کلمه mat
    - باید کلمات sat و vat یک مکان به راست شیفت داده شوند.
  - برای حذف کردن کلمه cat
    - باید کلمات sat و vat یک مکان به چپ شیفت داده شوند.

- مشکلات بازنمایی ترتیبی (آرایه)

۱- حذف و درج عناصر در آرایه ها بسیار وقت گیر است

۲- باذخیره کردن هر لیست در آرایه ای با حداکثر اندازه ، حافظه هدر می رود

S.Najjar.G@Gmail.com

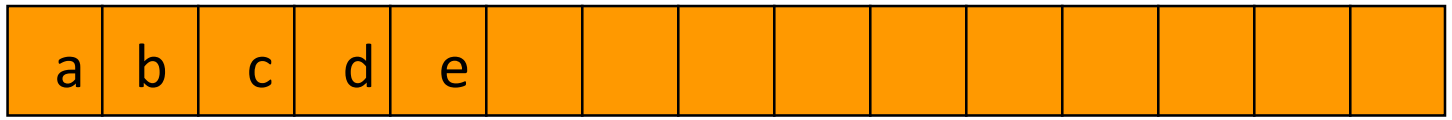
بازنمایی پیوندی (لیست پیوندی)

- راه حل مناسب : استفاده از بازنمایی پیوندی (لیست پیوندی)
- عناصر می‌توانند در هر جای حافظه قرار گیرند.
- در بازنمایی ترتیبی (آرایه)، ترتیب اعضای لیست با ترتیب نگهداری اعضا در حافظه یکسان است ولی در بازنمایی پیوندی لازم نیست ترتیب اعضای لیست با ترتیب نگهداری اعضا یکسان باشد.
- برای دستیابی صحیح به عناصر یک لیست ، بایستی به همراه هر عنصر، آدرس یا موقعیت عنصر بعدی نیز ذخیره شود.
- بنابراین برای هر عنصر لیست، یک نود وجود دارد که حاوی فیلدهای داده ای و اشاره گری به عنصر بعدی در لیست می باشد.

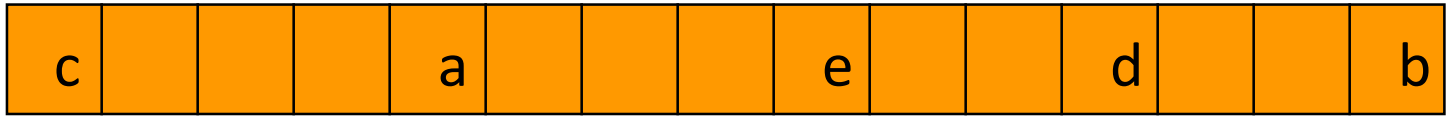
S.Najjar.G@Gmail.com

شمای حافظه

- شمای حافظه برای لیست L = (a,b,c,d,e)
  - بازنمایی ترتیبی (آرایه)

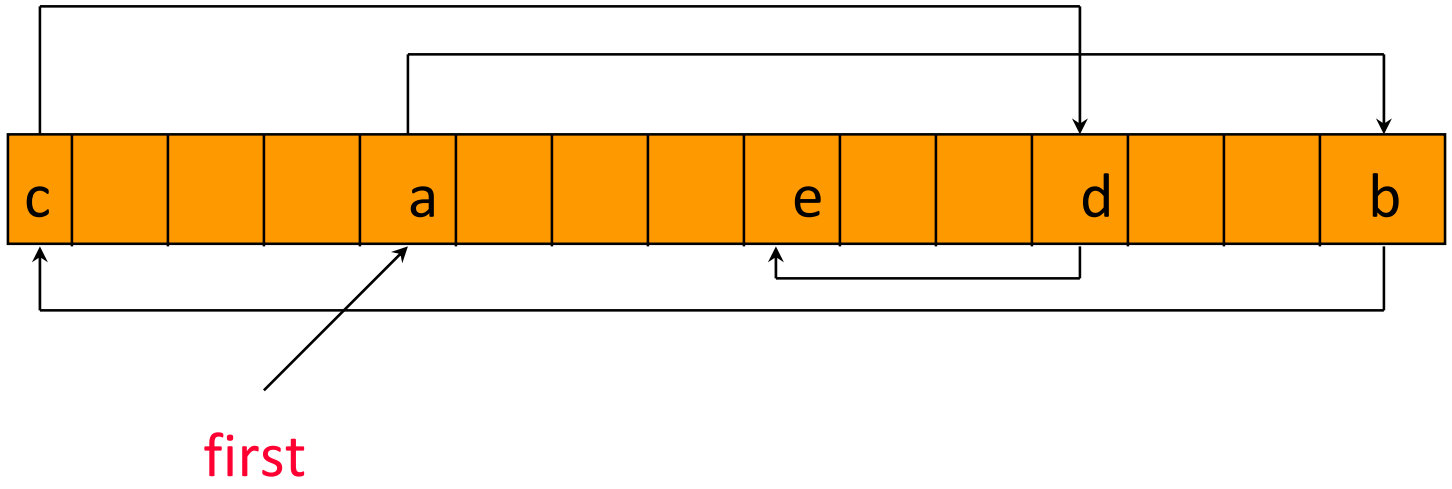


  - لیست پیوندی



S.Najjar.G@Gmail.com

لیست پیوندی



اشاره گر در e برابر NULL است.

از متغیر first برای دسترسی به عنصر اول استفاده می شود.

S.Najjar.G@Gmail.com

اشاره گرها

- C به صورت مناسبی از اشاره گرها حمایت می کند.
  - دو عملگری که با اشاره گرها به کار می روند:

& the address operator

* the dereferencing (or indirection) operator

مثال

اگر تعریف زیر را داشته باشیم

```
int i, *pi;
```

آنگاه iیک متغیر صحیح و pi یک اشاره گر به یک متغیر صحیح است.

و

```
pi = &i;
```

آدرس i را برگردانده و به عنوان مقدار pi نسبت می دهد.. برای مقدار دهی به i

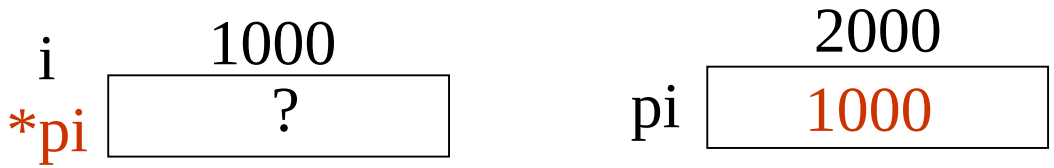
```
i = 10; or *pi = 10;
```

Pointer Review (1)

S.Najjar.G@Gmail.com

int i, *pi;

pi = &i;

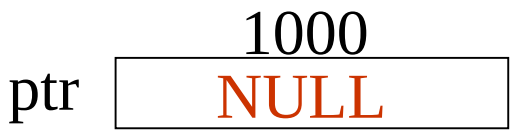


i = 10 or *pi = 10

Pointer Review (2)

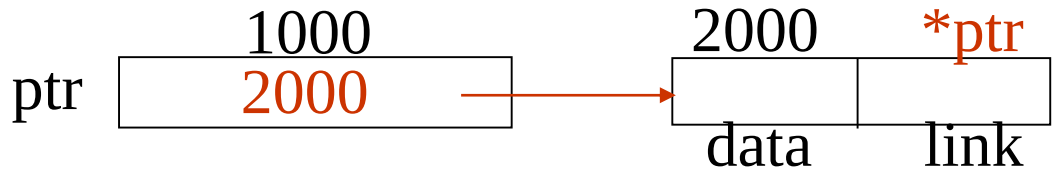
S.Najjar.G@Gmail.com

```
typedef struct list_node {
                    int data;
                    list_node *link;
                  }
list_node *ptr = NULL;
```



ptr->data⇨(*ptr).data

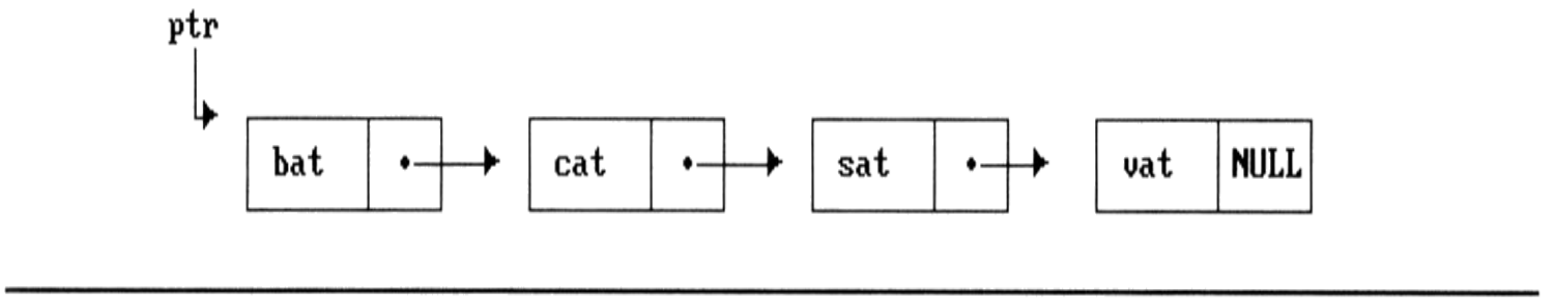
```
ptr = malloc(sizeof(list_node));
```



S.Najjar.G@Gmail.com

لیست های تک پیوندی

- لیست های پیوندی معمولا به وسیله گره هایی متوالی با اتصالاتی که به صورت فلش هایی نشان داده شده اند ارایه می گردند.



S.Najjar.G@Gmail.com

لیست های تک پیوندی

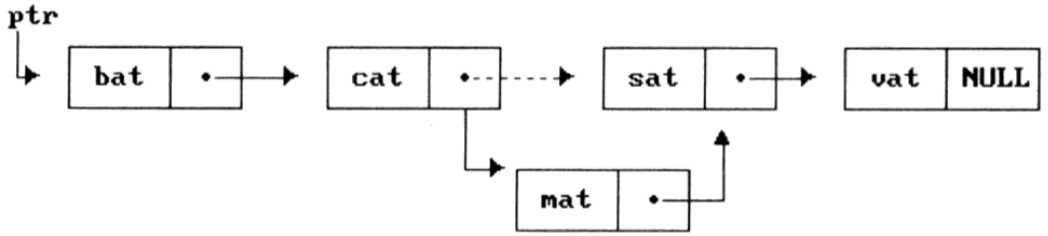
- برای اضافه کردن کلمه mat بین cat و sat :

۱- گره ی استفاده نشده ای را در نظر گرفته ، فرض کنید که آدرس آن paddr باشد.

۲- فیلد داده این گره را برابر با mat قرار دهید

۳- فیلد اتصال paddr را طوری تنظیم کنید که به ادرسی که در فیلد اتصال گره حاوی cat می باشد، اشاره کند

۴- فیلد اتصال گره حاوی cat را طوری تنظیم کنید که به paddr اشاره کند.



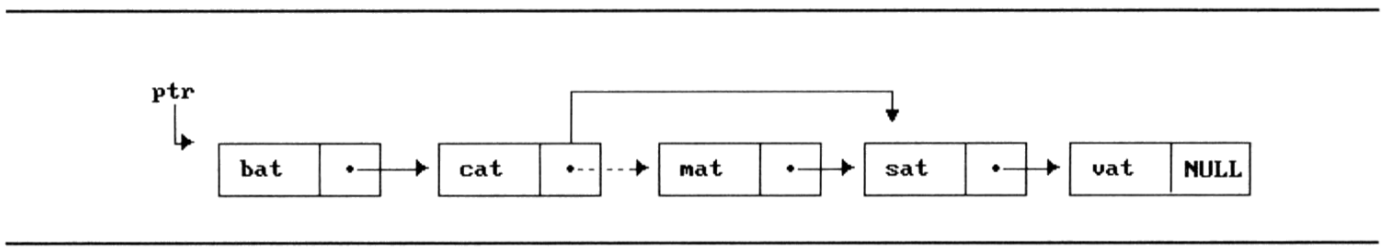
S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- حذف mat از لیست

✓ برای انجام این کار فقط لازم است که عنصر قبل از mat یعنی cat را پیدا و فیلد اتصال آنرا طوری تنظیم کنیم که به گره ای اشاره کند که در حال حاضر اتصال گره mat به ان اشاره دارد

✓ ما هیچ داده ای را جابجا نکرده ایم و با وجود آنکه فیلد اتصال mat هنوز به sat اشاره می کند matدیگر عضو لیست نیست.



S.Najjar.G@Gmail.com

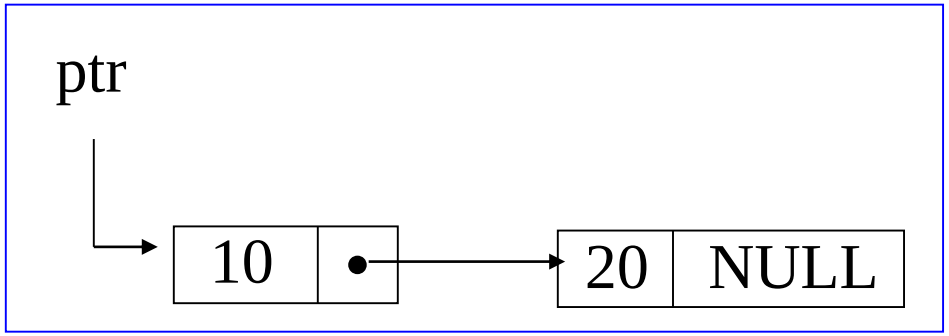
لیست های تک پیوندی

- **Example** [*Two-node linked list*]:

```
typedef struct list_node {
    int data;
     list_node *link;
};
 list_node *ptr =NULL;
```

- **Program :** Create a two-node list

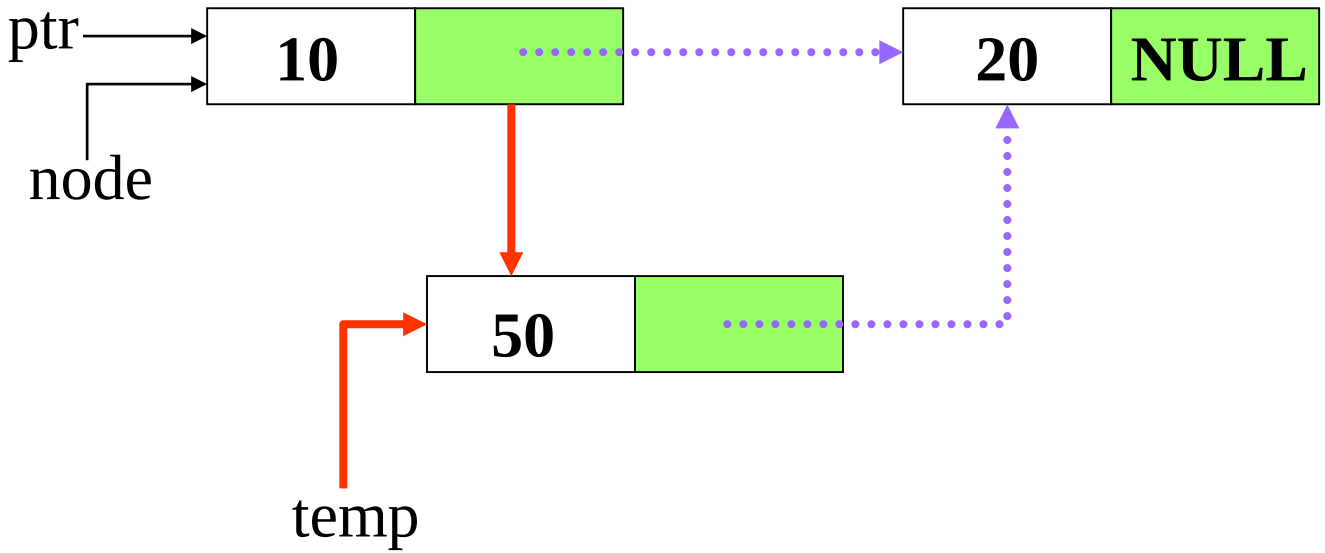
```
 list_node create2( )
{
    /* create a linked list with two nodes */
     list_node *first, *second;
    first = (list_node *) malloc(sizeof(list_node));
    second = (list_node *) malloc(sizeof(list_node));
    second -> link = NULL;
    second -> data = 20;
    first -> data = 10;
    first ->link = second;
    return first;
}
```



S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- اضافه کردن
  - یک نود با داده ۵۰ به لیست ptr و بعد از node اضافه کنید.

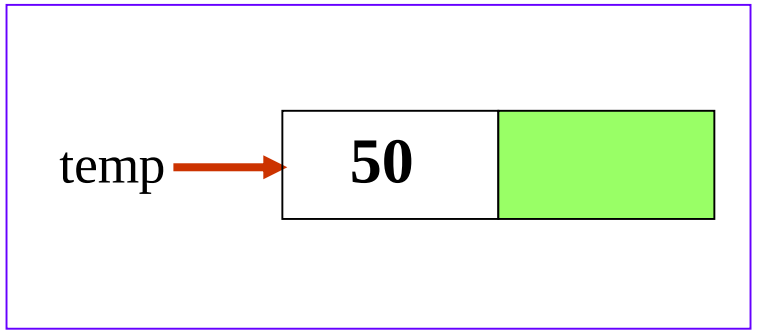


S.Nafar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- Implement Insertion:

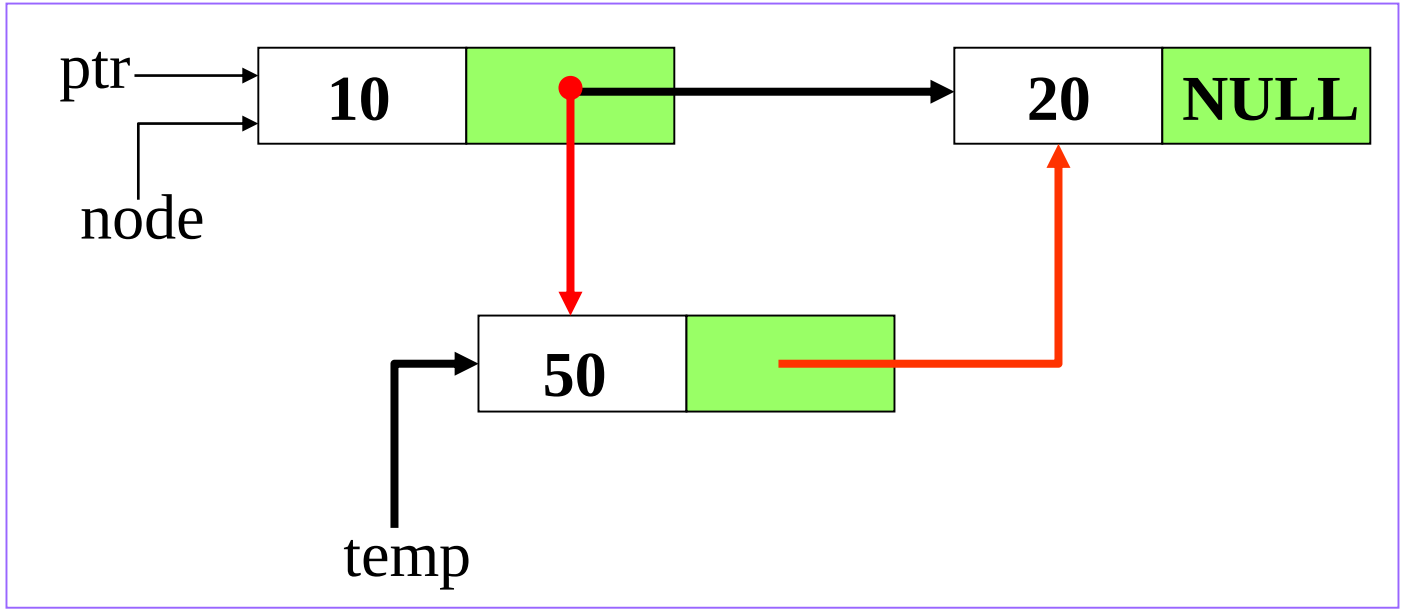
```
void insert(list_node *ptr, list_node *node)
{
/* insert a new node with data = 50 into the list ptr after node */
  list_node *temp;
  temp=(list_node *)malloc(sizeof(list_node));
  if(IS_FULL(temp)){
     fprintf(stderr, "The memory is full\n");
     exit(1);
  }
  temp->data=50;
```



SNajjarG@Gmail.com

لیست های تک پیوندی (ادامه)

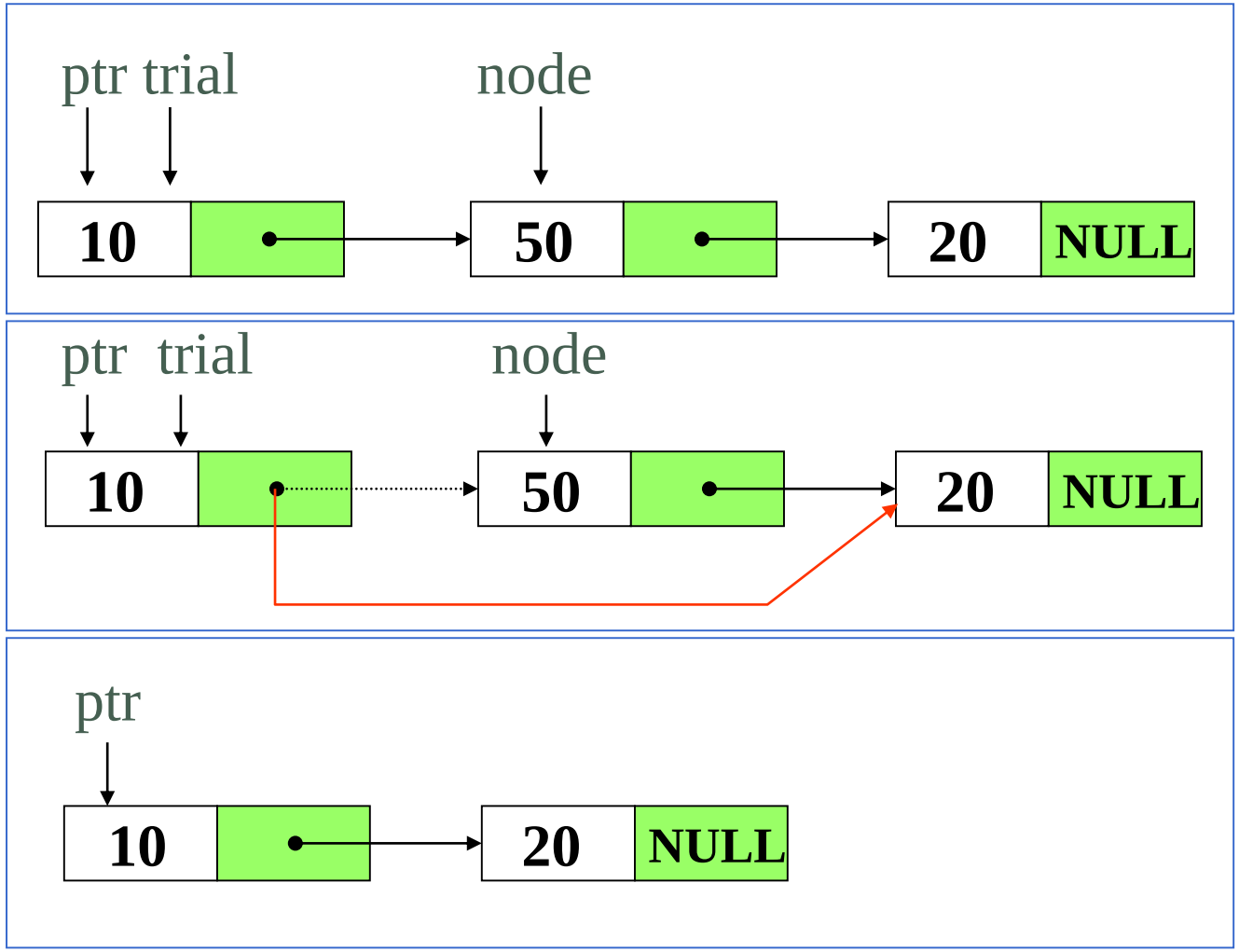
```
    if(*ptr){      //nonempty list
          temp->link = node->link;
          node->link = temp;
    }
    else{          //empty list
          temp->link = NULL;
          *ptr = temp;
    }
}
```



S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- حذف کردن عضوی که node به آن اشاره می کند را حذف کنید

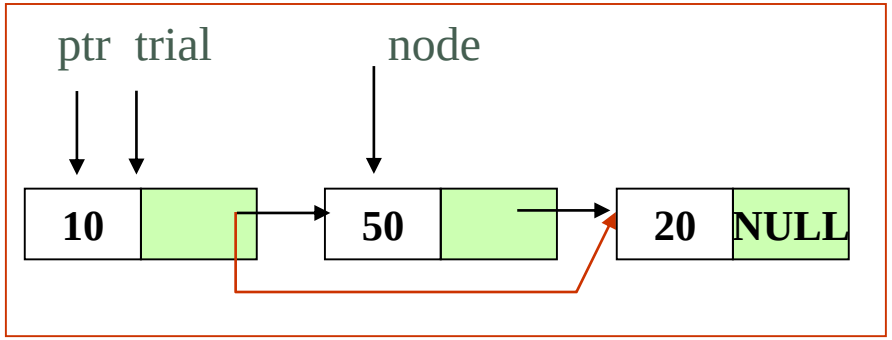


S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- Implement Deletion:

```
void delete(list_node *ptr, list_node *trail, list_node *node)
{
/* delete node from the list, trail is the preceding node ptr is
   the head of the list */
   if(trail<>Null)
      trail->link = node->link;
   else
      *ptr = (*ptr)->link;
   free(node);
}
```

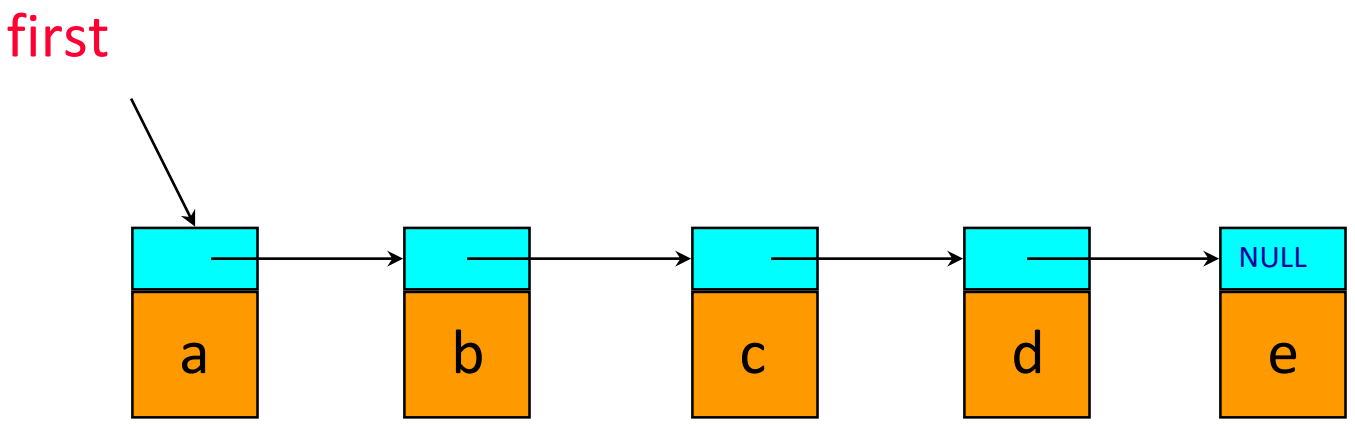


S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- پیمایش (چاپ) یک لیست پیوندی

Get(0)



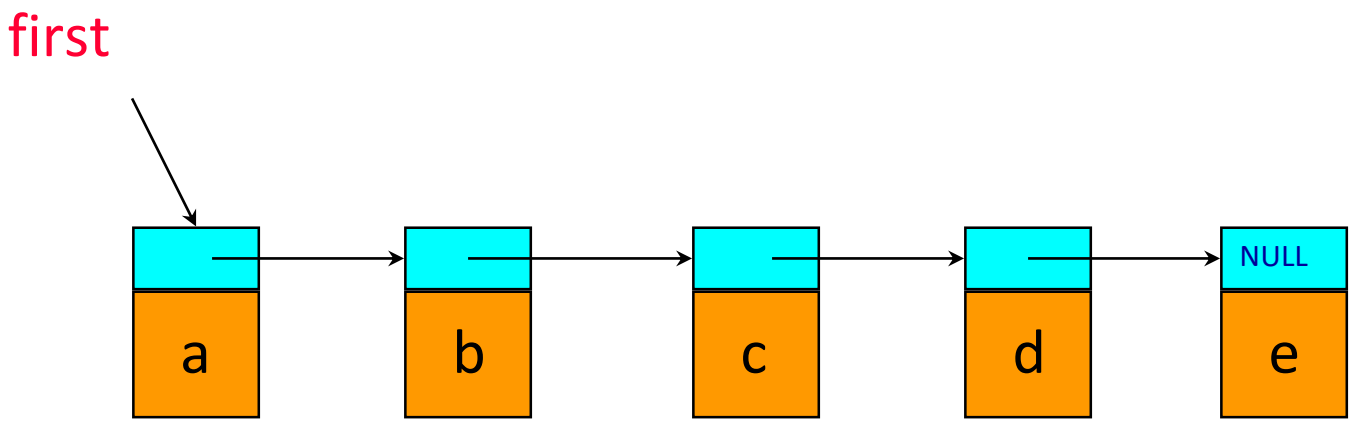

```
desiredNode = first; // gets you to first node
return desiredNode–>data;
```

S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- پیمایش (چاپ) یک لیست پیوندی

Get(1)



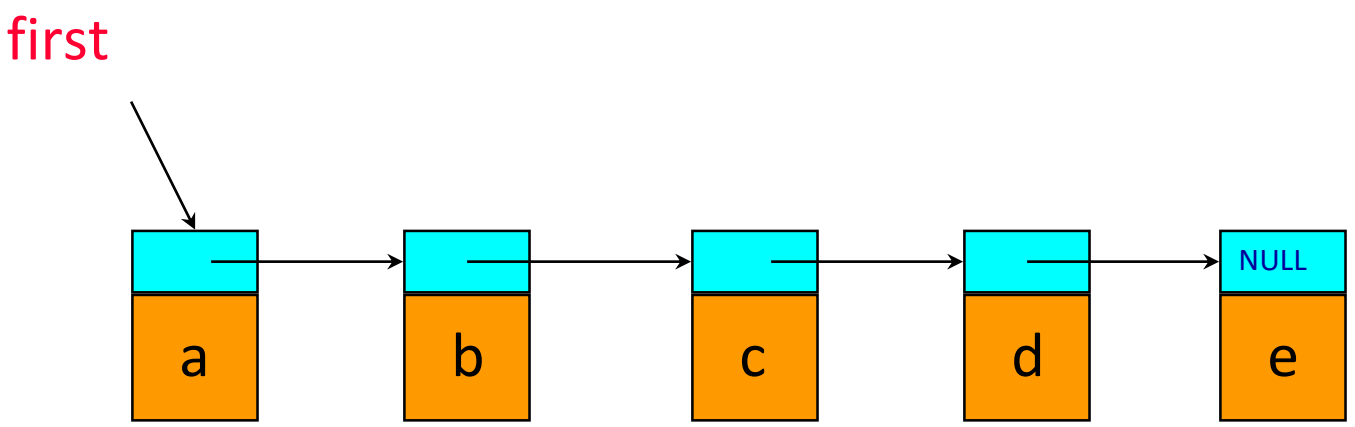

```
desiredNode = first–>link; // gets you to second node
return desiredNode–>data;
```

S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- پیمایش (چاپ) یک لیست پیوندی

Get(2)



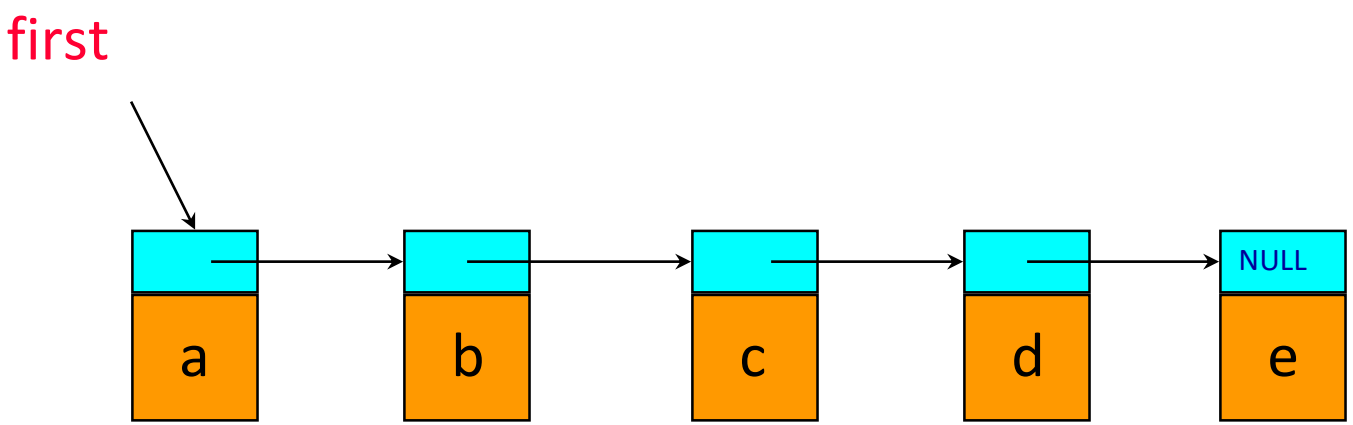

```
desiredNode = first–>link–>link; // gets you to third node
return desiredNode–>data;
```

S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- پیمایش (چاپ) یک لیست پیوندی

Get(5)




```
desiredNode = first–>link–>link–>link–>link–>link; // desiredNode = NULL
return desiredNode–>data;   // NULL.element
```

S.Najjar.G@Gmail.com

لیست های تک پیوندی (ادامه)

- پیمایش (چاپ) یک لیست پیوندی

- **Program :** Printing a list

```
void print_list(list_pointer first)
{
   printf(“The list contains: “);
   for ( ; first<>null; first= first->link)
   printf(“%4d”, first->data);
   printf(“\n”);
}
```

S.Najjar.G@Gmail.com

عملیات روی لیست‌های پیوندی

- اضافه کردن یک گره به انتهای لیست پیوندی

```
void attach(list_node *first, list_node *last, list_node *newnode)
{
   if(first==0) first=last=newnode;
   else
   {
       last->link=newnode;
      last=newnode;
   }
}
```

فرض می کنیم عضو داده ای last وجود دارد که به گره آخر لیست پیوندی اشاره می کند

S.Najjar.G@Gmail.com

عملیات روی لیست های پیوندی (ادامه)

- اتصال دو زنجیر

```
void Concatenate(list_node *first_a, list_node *first_b)
{
    if (!first_a) { first_a=first_b; return;}
   if (first_b) {
        for (list_node p=first; p->link; p=p->link); no body
         p->link=first_b;
    }
}
```

زمان اجرا بر حسب طول زنجیر اول خطي است

S.Najjar.G@Gmail.com

عملیات روی لیست های پیوندی (ادامه)

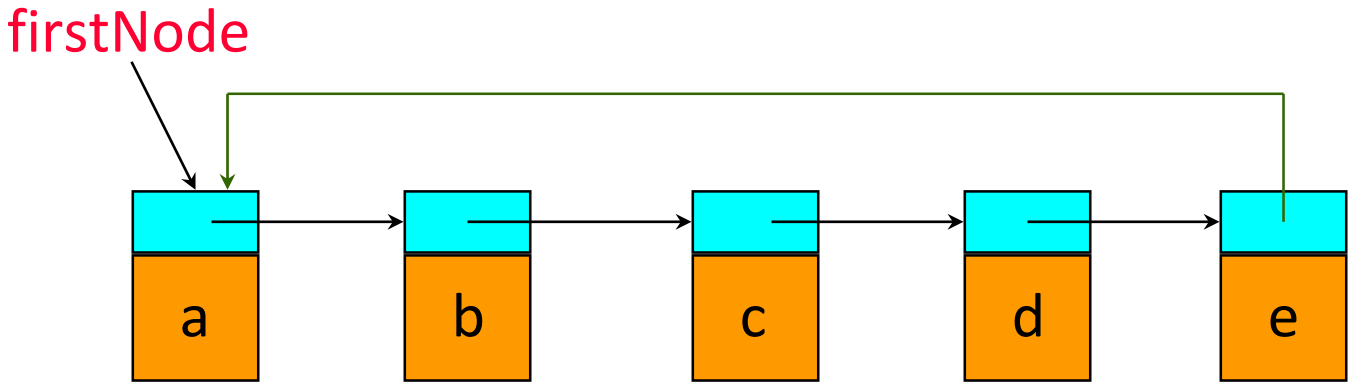
- معکوس کردن لیست پیوندی

```
void Invert(list_node *first)
{
 list_node p=first, q=0, r=0;     //q trails p
    while( p) {
          r=q; q=p;                //r trails q
          p=p->link;                 //p moves to next node
          q->link=r;                 //link q to preceding node
    }
    first =q;
}
```

o(m) زمان اجرا mبراي ليست به طول

S.Najjar.G@Gmail.com

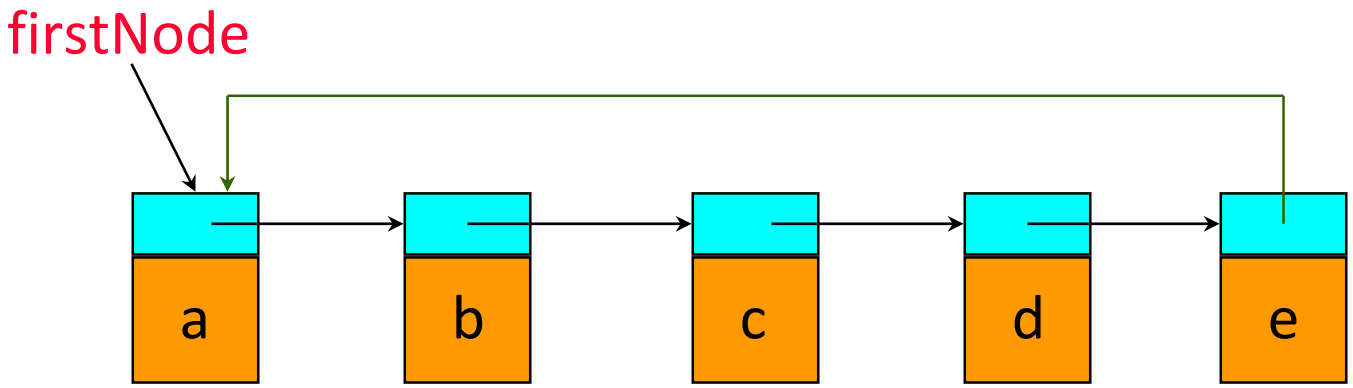
لیست دایره ای



- برای بررسی اینکه ایا اشاره گر current به گره آخر لیست دایره ای اشاره می کند به جای مقایسه (current->link==0) مقایسه ی (current->link==first) انجام می شود.
- الگوریتم های حذف/ اضافه کردن از/ به لیست دایره ای باید تضمین کند که فیلد اشاره گر گره آخر به گره اول لیست اشاره کند.

S.Najjar.G@Gmail.com

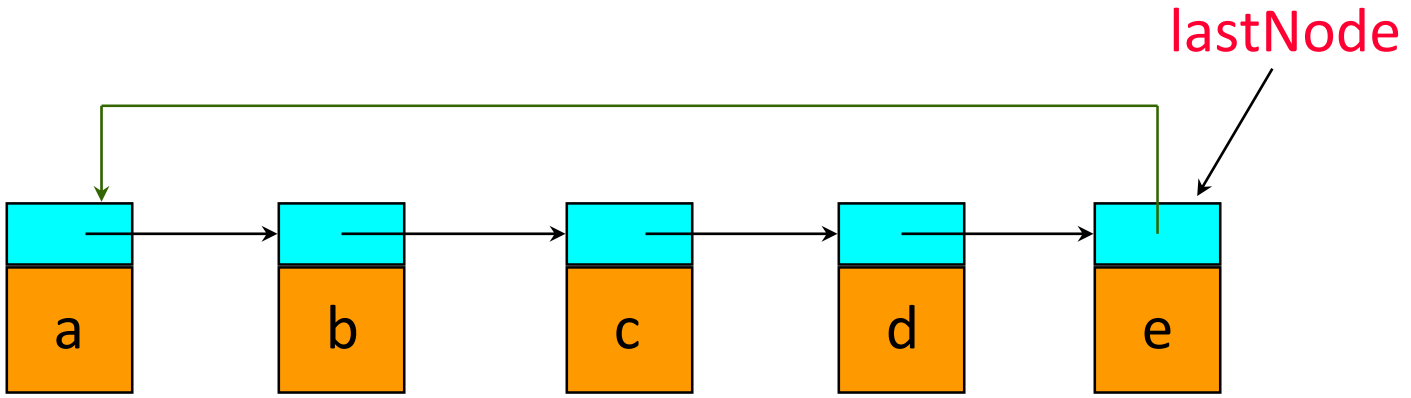
لیست دایره ای (ادامه)



- می خواهیم گره جدیدی به اول لیست بالا اضافه کنیم
  - باید اشاره گر گره آخر (گره ای که حاوی e است) را تغییر دهیم.
  - باید تا پیدا نشدن گره آخر در طول لیست حرکت کنیم.

S.Najjar.G@Gmail.com

لیست دایره ای (ادامه)



- هنگامی که از گره سر استفاده نمی کنیم بهتر است اشاره گر دسترسی به لیست دایره ای به جای گره اول به گره آخر لیست اشاره کند.
- در اینصورت اضافه کردن یک گره در اول و یا در آخر لیست دایره ای در مدت زمان ثابتی انجام می شود.

S.Najjar.G@Gmail.com

لیست دایره ای (ادامه)

- اضافه کردن گره ای که x به آن اشاره می کند در اول لیست

```
void InsertFront(list_node *last, list_node *x)
// insert the node pointed at by x at the front of the circular list
// last points to the last node in the list
{
   if (!last) {  // empty list
        last=x; x->link=x;
   }
   else {
        x->link=last->link;
        last->link=x;
   }
}
```

زمان اجرا O(1) است

S.Najjar.G@Gmail.com

لیست دایره ای (ادامه)

- اضافه کردن گره ای که x به آن اشاره می کند در انتهای لیست

```
void InsertRear(list_node *last, list_node *x)
// insert the node pointed at by x at the rear of the circular list
// last points to the last node in the list
{
    if (!last) {  // empty list
         last=x; x->link=x;
    } else {
        x->link=last->link; last->link=x;
        last=x;
    }
}
```

تفاوت با کد قبلی

زمان اجرا O(1) است

Home Work 9

S.Najjar.G@Gmail.com

- پیاده‌سازی توابع حذف، درج و پیمایش لیست پیوندی و اجرا یک مثال در آن.