به نام خدا

ساختمان داده ها

جلسه هشتم

دانشگاه صنعتی همدان

گروه مهندسی کامپیوتر

نیم سال دوم 98-1397

فصل چهارم

لیستهای پیوندی و کاربرد آنها

Linked Lists

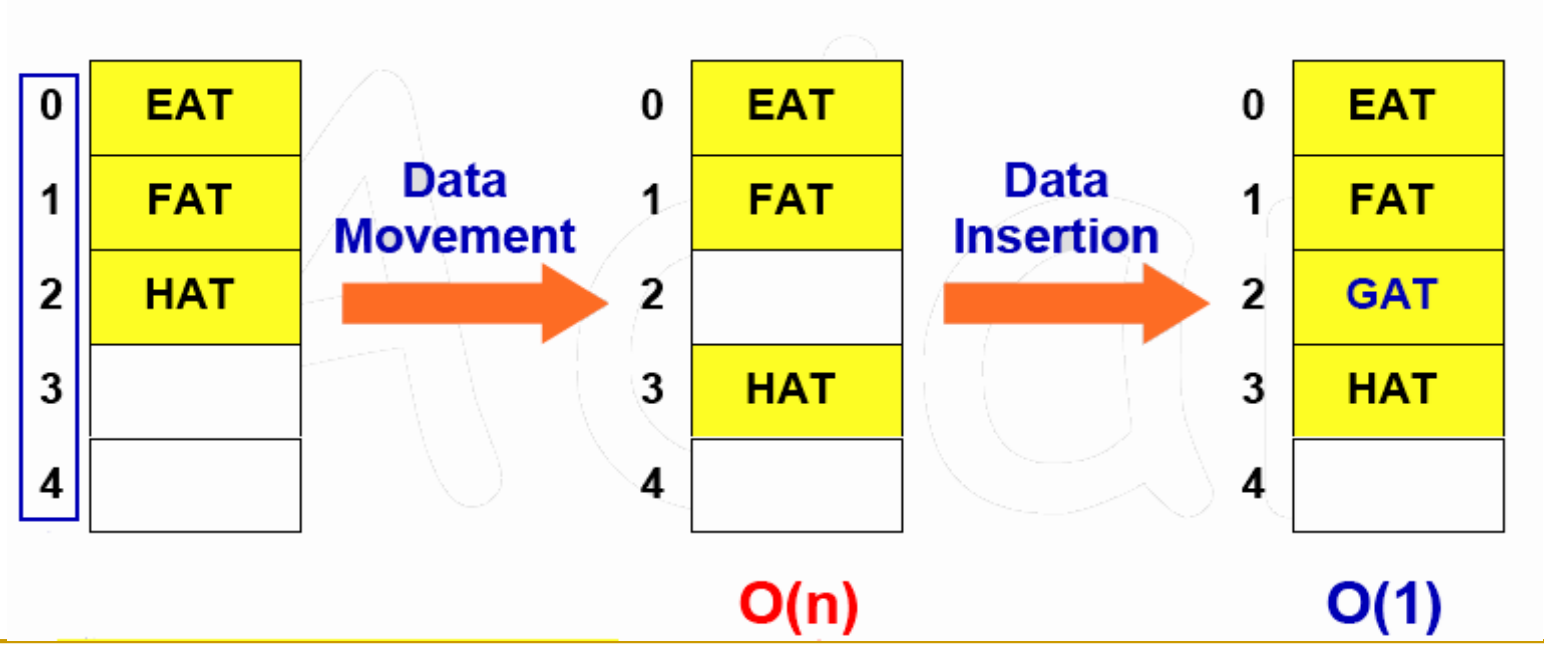
لیستهای تک پیوندی

- در قسمتهای گذشته برای پیاده سازی ساختمان داده های مورد نظر از آرایه استفاده می کردیم. خاصیت ارایه ها این بود که عناصر پشت سر هم ذخیره می شدند. بنابر این خواص زیر:
  - اگر عنصر a_{ij} در موقعیت L_{ij} ذخیره شده باشد عنصر $a_{i,j+1}$ در موقعیت $L_{i,j+c}$ قرار می گیرد.
  - اگر i امین گره در صف در موقعیت L_i باشد انگاه در صف حلقوی گره (i+1) در موقعیت $(L_i+c)\%n$ خواهد بود.
  - اگر در پشته بالاترین گره در موقعیت L_T باشد انگاه گره پایین تر در موقعیت L_T-c خواهد بود

لیستهای تک پیوندی

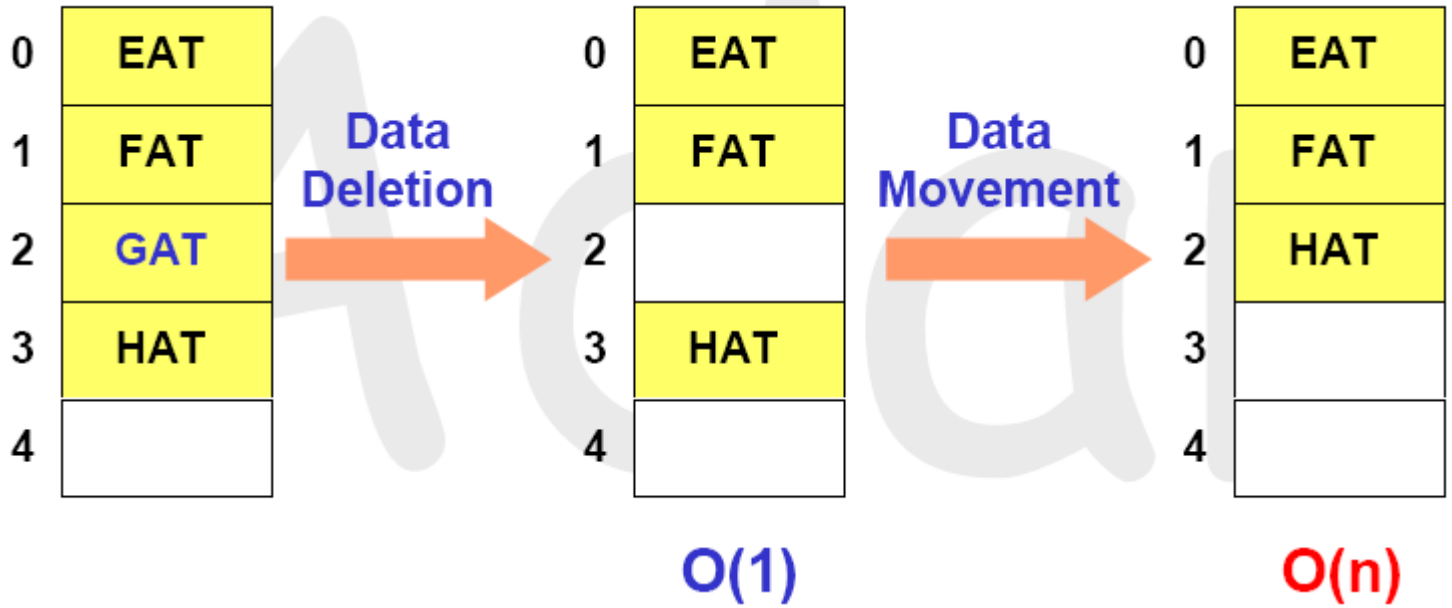
این خواص نشان می دهد که این ساختار برای بسیاری از عملها مانند درج و یا حذف عناصر از یک پشته یا صف مناسب است. ولی اگر در یک لیست ترتیبی بخواهیم این اعمال را انجام بدهیم کار بسیار زمان بری است . O(1) و O(n)

مثال اضافه کردن کلمه GAT در یک لیست مرتب.



لیستهای تک پیوندی

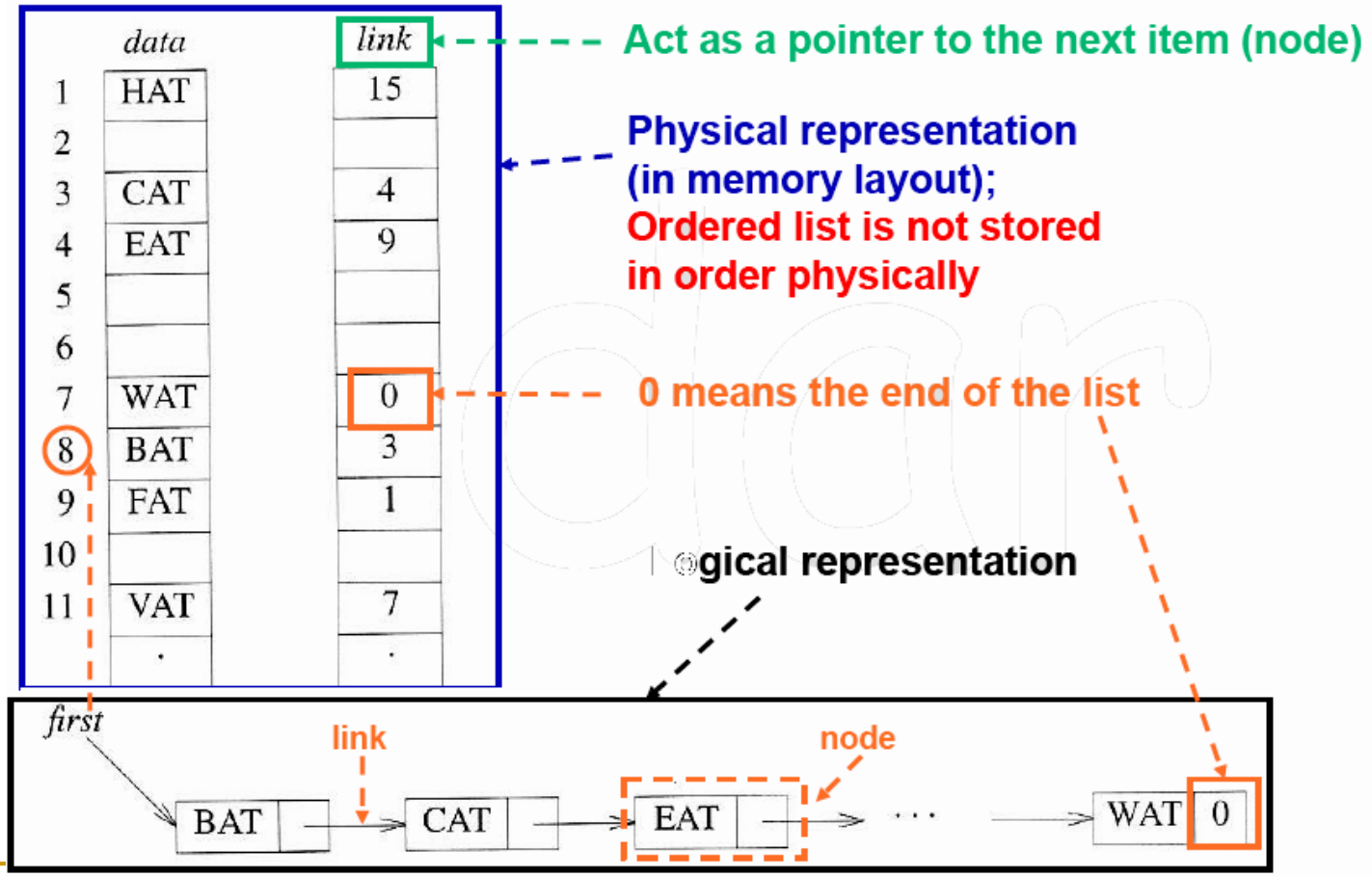
مثال حذف از یک لیست مرتب



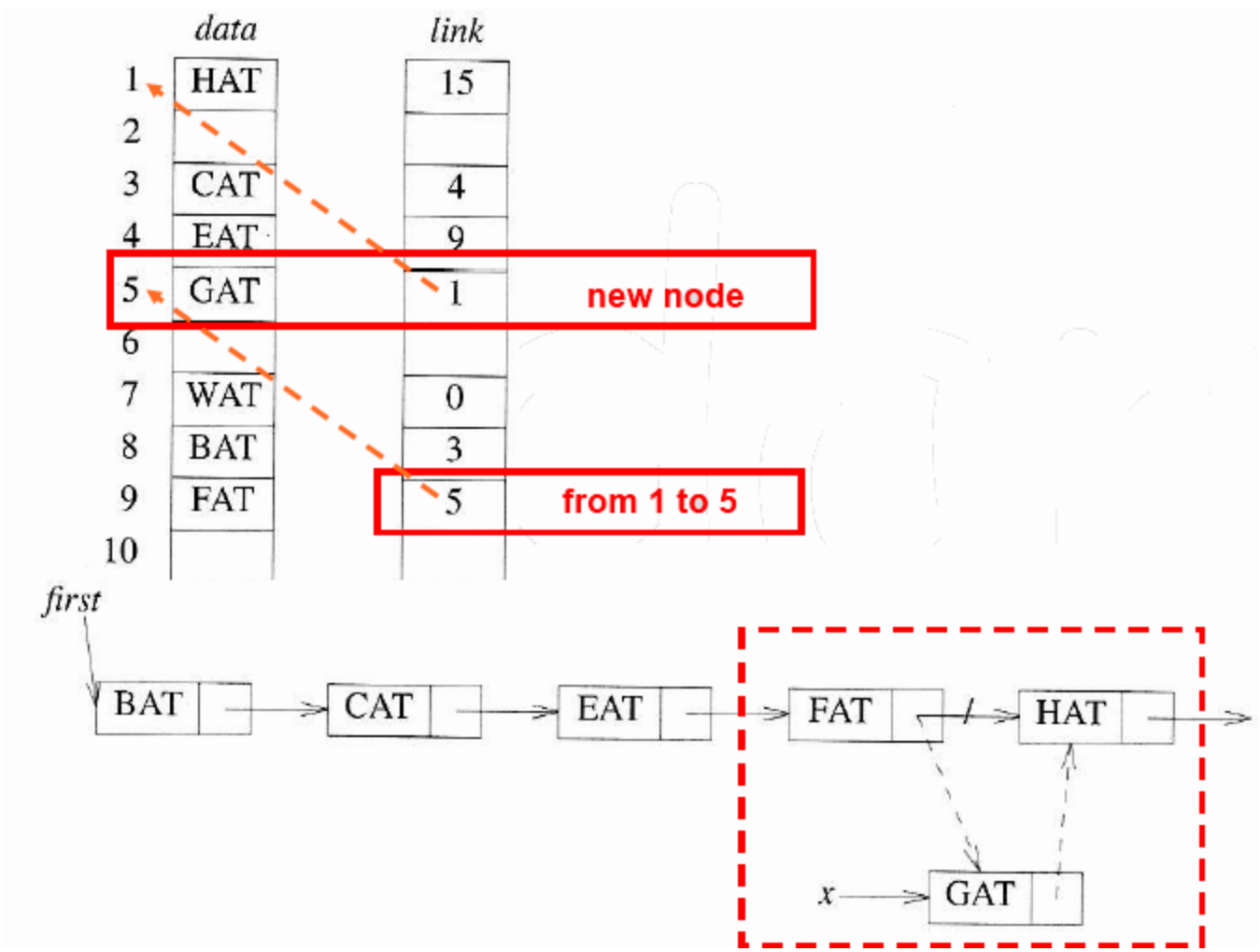
مشکل دیگری که در ارتباط با این ساختار و جود دارد ثابت بودن اندازه آرایه است و باید هنگام تعریف آن اندازه آنرا نیز مشخص کرد. باعث می شود که گاهی وقتها پشته یا صف پر شوند و جای خالی موجود نباشد.

لیتهای تک پیوندی

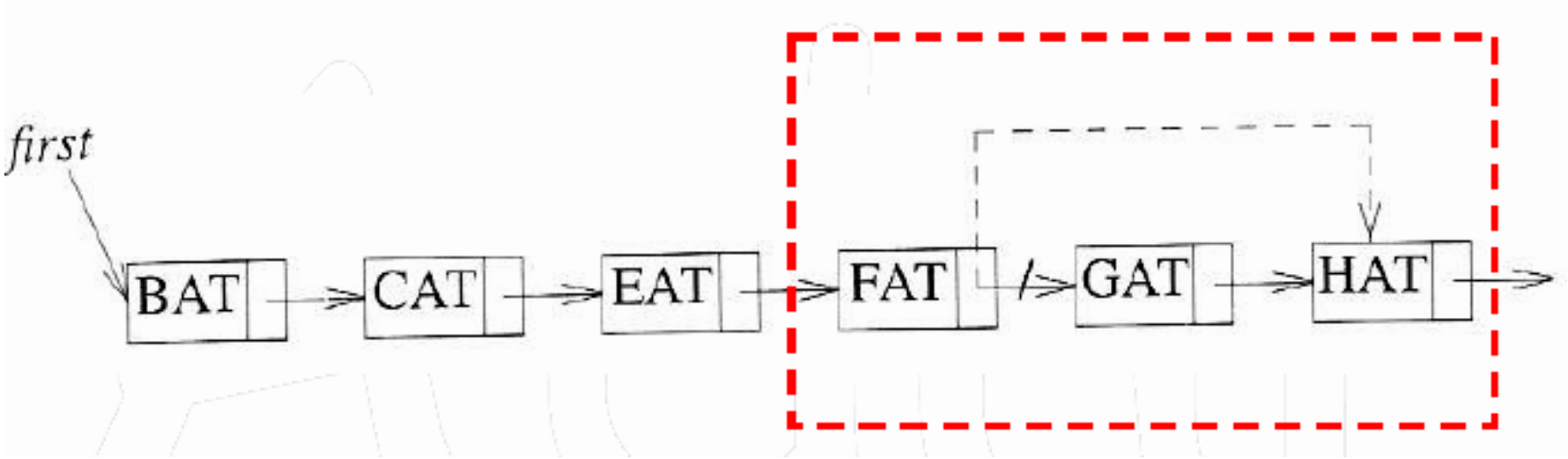
- راه حل مناسب برای رفع این مشکلات استفاده از لیستهای پیوندی می باشد.



اضافه کردن یک عنصر به لیست



حذف از یک لیست



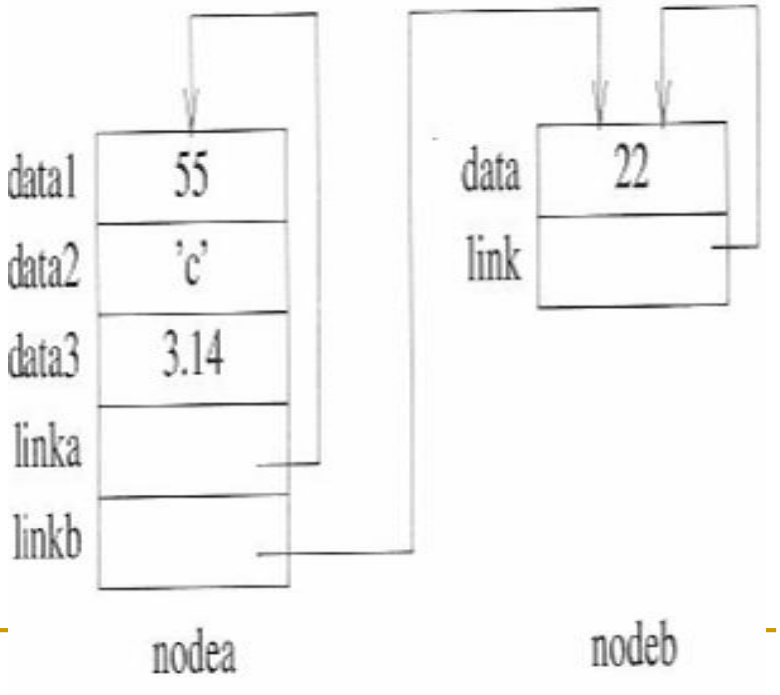
نمایش لیستها در C++

گره یا Node ساختمان داده ای است که از یک فیلد داده ای و یک فیلد اشاره گری که به گره دیگری اشاره میکند تشکیل شده است. مثلا برای مثال قبل تعریف گره می تواند به صورت زیر باشد:

```
class ThreeLetterNode
{
char data[3];
ThreeLetterNode*    link;
};
```

می توان ساختار پیچیده تری از لیستها نیز به صورت زیر تعریف نمود:

```
class nodeb
{
int data;
nodeb *link;
};
class nodea
{
Int  data1; char  data2;
Float  data3;
nodea*   linka;
nodeb*   linkb;
};
```



طراحی ساختمان داده یک لیست

روش اول : در این روش متغیر سراسری first را به صورت سراسری تعریف می کنیم:

```
ThreeLetterNode*   first;
```

در این حالت برای دسترسی به عناصر این گره داریم

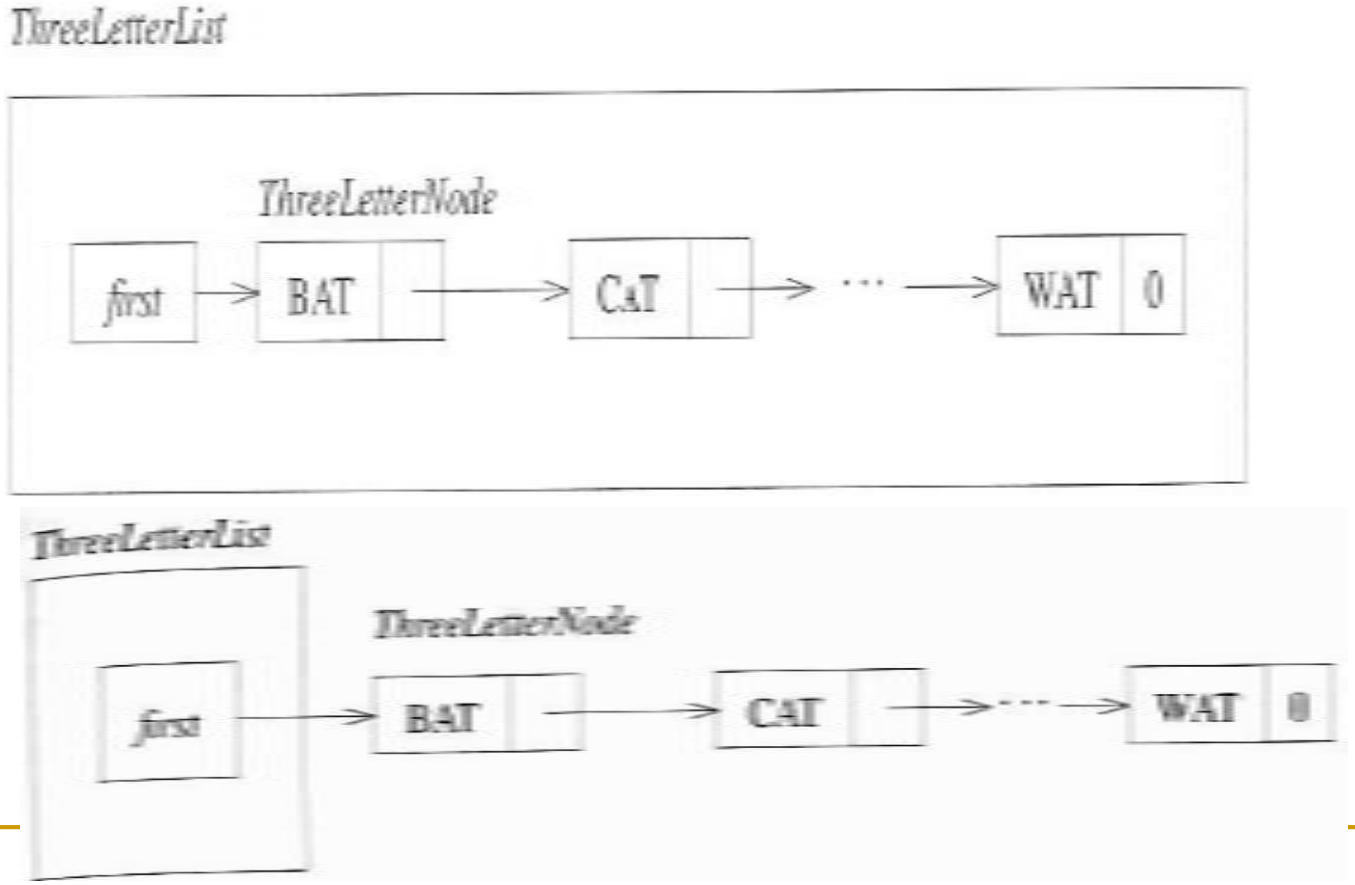
```
First->data;  ,  first->link;
```

مشکلی که این روش دارد دسترسی به متغیرهای خصوصی است که باعث به وجود آمدن خطا خواهد شد.

روش دوم: برای رفع این مشکل می توان داده ها را به صورت عمومی تعریف کرد یا توابعی برای دسترسی به آنها تعریف نمود که اصل پنهان سازی را با اشکال مواجه می کند.

طراحی ساختمان داده یک لیست

- روش سوم: در این روش یک کلاس برای کل لیست پیوندی تعریف می کنیم که توابعی را که برای دستکاری لیست لازم است پشتیبانی می کند. در واقع این کلاس اشاره گری به اولین گره لیست خواهد بود.



تعریف کلاس لیست به صورت ترکیبی

```
class ThreeLetterList; // forward declaration

class ThreeLetterNode
{
Friend  class ThreeLetterList;
private:
char data[3];
ThreeLetterNode*    link;
}


class ThreeLetterList
{
private:
ThreeLetterNode*   first;

public:       // list manipulation functions, discussed later
};
```

تعریف کلاس لیست به صورت تو در تو

```
class ThreeLetterList
{
private:
class ThreeLetterNode
{
public:
char data[3];
ThreeLetterNode*    link;
}
ThreeLetterNode*   first;

public:       // list manipulation functions, discussed later
};
```

اعمال دستکاری لیستها

```
class List; // forward declaration
class ListNode
{
friend class List;
int    data;
ListNode*  link;
public:
ListNode(int  value = 0);
}

class List
{
ListNode*  first;
public:
void Create2();
void Insert50(ListNode *x);
void Delete(ListNode*x, ListNode*y);
};
```

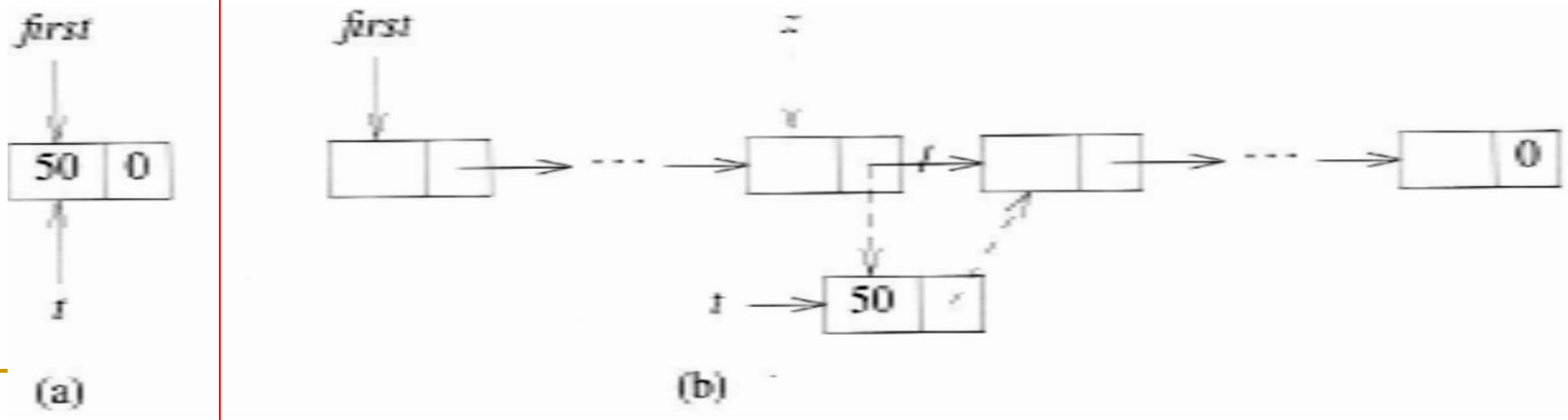
```
ListNode::ListNode(int value):data(value), link(0)
{}

void List::Create2()
{

first = new ListNode(10);
first->link= new ListNode(20);

}
```

```
void List::Insert50(ListNode *x)
 {
ListNode* t = new ListNode(50);
If   (first== 0)
{
first = t;
return;
}
t->link = x->link;
x->link = t;
}
```



```
void List::Delete(ListNode*x, ListNode*y)
{
If  (! y)
first = first->link;
else
y->link = x->link;
delete x;
}
```