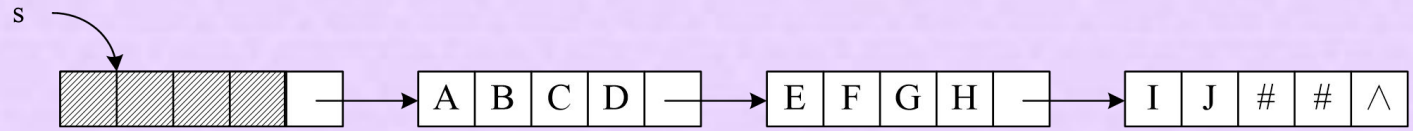
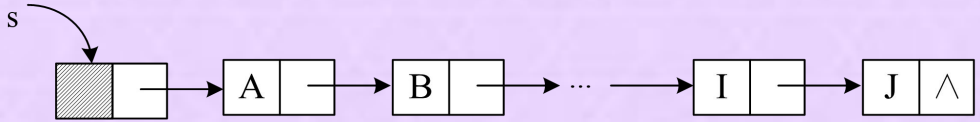
本节主题：

串的链式存储及其基本操作实现

串的链式存储

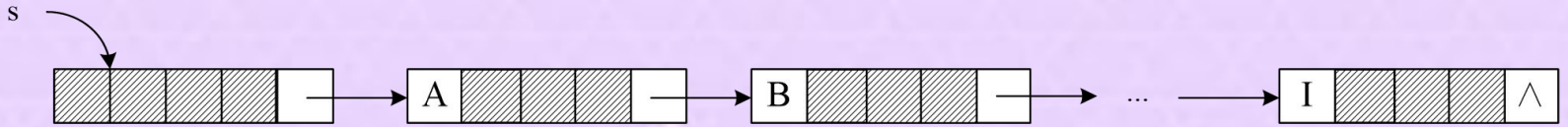


数据域以“字”为单位的链串——存储密度大



数据域以“字节”为单位的链串——存储密度小

```
typedef struct snode
{
    char data;
    struct snode *next;
} LiString;
```



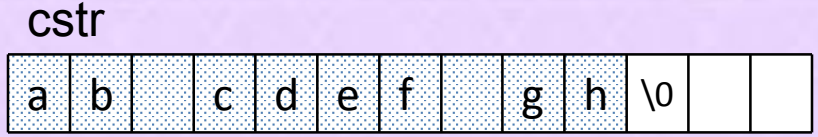
“数据对齐”影响下，C语言程序中实际的链表

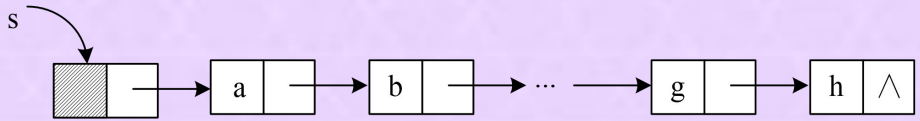
串赋值：StrAssign(s,cstr)

```
void StrAssign(LiString *&s,char cstr[])
{
    int i;
    LiString *r,*p;
    s=(LiString *)malloc(sizeof(LiString));
    r=s;        //r始终指向尾节点
    for (i=0; cstr[i]!='\0'; i++)
    {
        p=(LiString *)malloc(sizeof(LiString));
        p->data=cstr[i];
        r->next=p;
        r=p;
    }
    r->next=NULL;
}
```

将一个字符串常量cstr赋给串s

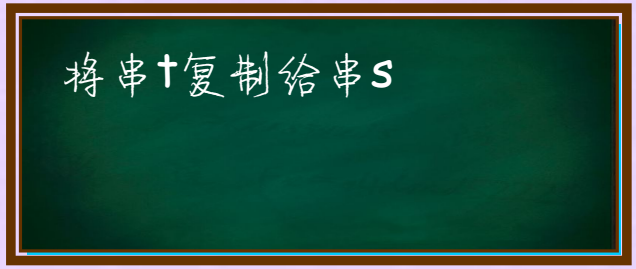
方法：尾插法

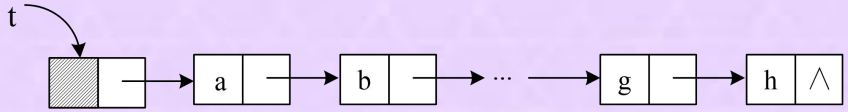


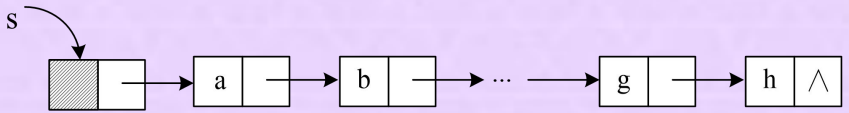


串复制：StrCopy(s,t)

```
void StrCopy(LiString *&s,LiString *t)
{
   LiString *p=t->next,*q,*r;
   s=(LiString *)malloc(sizeof(LiString));
   r=s;        //r始终指向尾节点
   while (p!=NULL)  //将t的所有节点复制到s
   {
      q=(LiString *)malloc(sizeof(LiString));
      q->data=p->data;
      r->next=q;
      r=q;
      p=p->next;
   }
   r->next=NULL;
}
```



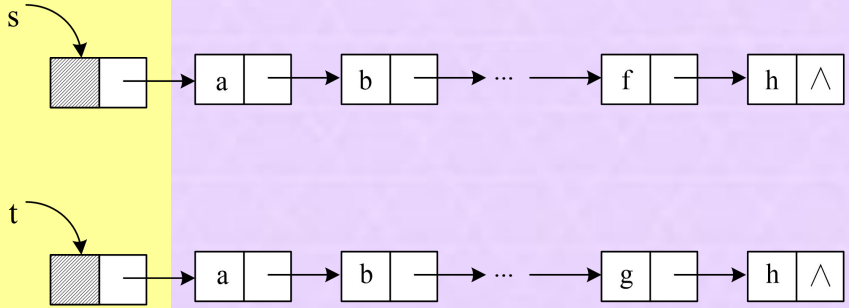




判串相等：StrEqual(s,t)

若两个串s与t相等返回真（1）；否则返回假（0）

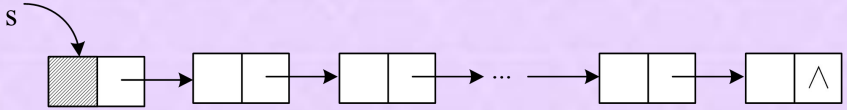
```
bool StrEqual(LiString *s,LiString *t)
{
   LiString *p=s->next,*q=t->next;
   while (p!=NULL && q!=NULL && p->data==q->data)
   {
      p=p->next;
      q=q->next;
   }
   if (p==NULL && q==NULL)
      return true;
   else
      return false;
}
```



求串长：StrLength(s)

返回串的长度。

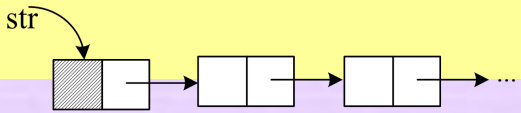
```
int StrLength(LiString *s)
{
    int i=0;
    LiString *p=s->next;
    while (p!=NULL)
    {
        i++;
        p=p->next;
    }
    return i;
}
```



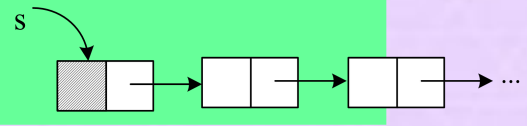
串连接：Concat(s,t)

将两个串s和t连接形成新串，返回这个新串。

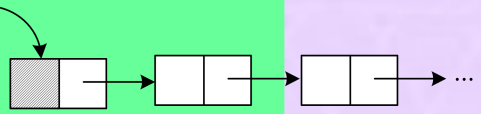
```
LiString *Concat(LiString *s,LiString *t)
{
    LiString *str,*p=s->next,*q,*r;
    str=(LiString *)malloc(sizeof(LiString));
    r=str;
    //将s的所有节点复制到str
    //将t的所有节点复制到str
    r->next=NULL;
    return str;
}
```




```
while (p!=NULL)
{
    q=(LiString*)malloc(sizeof(LiString));
    q->data=p->data;
    r->next=q;
    r=q;
    p=p->next;
}
```

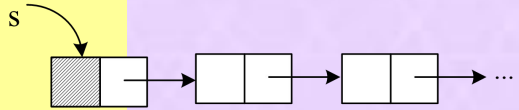



```
p=t->next;
while (p!=NULL)
{
    q=(LiString*)malloc(sizeof(LiString));
    q->data=p->data;
    r->next=q;
    r=q;
    p=p->next;
}
```



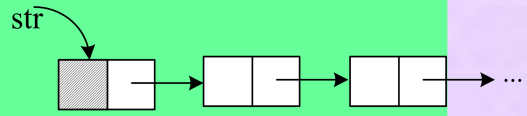
求子串: SubStr(s,i,j)

返回串s中从第i个字符开始的、由连续j个字符组成的子串。
参数不正确时返回一个空串。
（1≤i≤StrLength(s)）




```
LiString *SubStr(LiString *s,int i,int j)
{
   int k;
   LiString *str,*p=s->next,*q,*r;
   str=(LiString *)malloc(sizeof(LiString));
   str->next=NULL;
   r=str;     //r指向新建链表的尾节点
   if (i<=0 || i>StrLength(s) || j<0 || i+j-1>StrLength(s))
      return str;   //参数非法时的处理
   for (k=0; k<i-1; k++)
      p=p->next;
   //将s的第i个节点开始的j个节点复制到str
   r->next=NULL;
   return str;
}
```

```
for (k=1; k<=j; k++)
{
   q=(LiString *)malloc(sizeof(LiString));
   q->data=p->data;
   r->next=q;
   r=q;
   p=p->next;
}
```



插入串：InsStr(s,i,t)

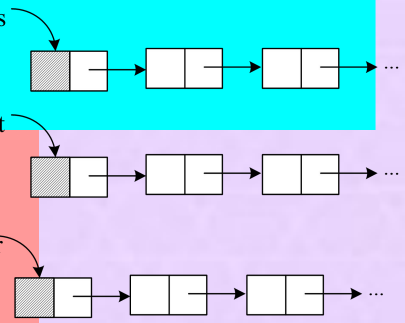
将串t插入到串s的第i个字符中，即将t的第一个字符作为s的第i个字符，并返回产生的新串。
参数不正确时返回一个空串。
（1≤i≤StrLength(s1)+1）

```
LiString *InsStr(LiString *s,int i,LiString *t)
{
   int k;
   LiString *str,*p=s->next,*p1=t->next,*q,*r;
   str=(LiString *)malloc(sizeof(LiString));
   //参数非法时的处理
   r=str;
   //将s的前i个节点复制到str
   //将t的所有节点复制到str
   //将*p及其后的节点复制到str
   r->next=NULL;
   return str;
}
```

```
for (k=1; k<i; k++)
{
   q=(LiString *)malloc(sizeof(LiString));
   q->data=p->data;
   r->next=q;
   r=q;
   p=p->next;
}
```

```
while (p1!=NULL)
{
   q=(LiString *)malloc(sizeof(LiString));
   q->data=p1->data;
   r->next=q;
   r=q;
   p1=p1->next;
}
```

```
while (p!=NULL)
{
   q=(LiString *)malloc(sizeof(LiString));
   q->data=p->data;
   r->next=q;
   r=q;
   p=p->next;
}
```

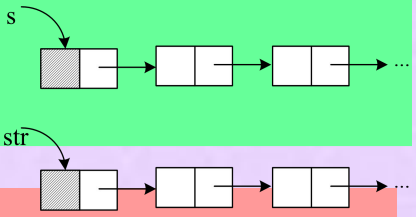


删除串中字符：DelStr(s,i,j)

从串s中删去第i个字符开始的长度为j的子串，并返回产生的新串。
参数不正确时返回一个空串，
（1≤i≤StrLength(s)）

```
LiString *DelStr(LiString *s,int i,int j)
{
   int k;
   LiString *str,*p=s->next,*q,*r;
   str=(LiString *)malloc(sizeof(LiString));
   //参数非法时的处理
   r=str;
   //将s的前i-1个节点复制到str
   //让p沿next跳j个节点
   //将*p及其后的节点复制到str
    r->next=NULL;
   return str;
}
```

```
for (k=0; k<i-1; k++)
{
   q=(LiString *)malloc(sizeof(LiString));
   q->data=p->data;
   r->next=q;
   r=q;
   p=p->next;
}
```




```
for (k=0; k<j; k++)
   p=p->next;
```

```
while (p!=NULL)
{
   q=(LiString *)malloc(sizeof(LiString));
   q->data=p->data;
   r->next=q;
   r=q;
   p=p->next;
}
```

替换子串：RepStr(s,i,j,t)

在串s中，将第i个字符开始的j个字符构成的子串用串t替换，并返回产生的新串。参数不正确时返回一个空串。（1≤i≤StrLength(s)）

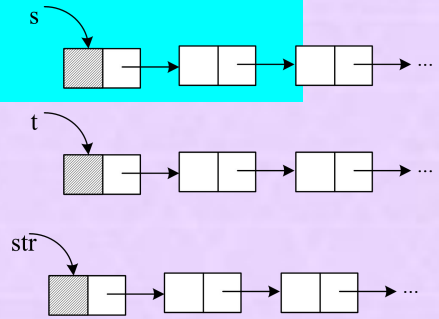
```
LiString *RepStr(LiString *s,int i,int j,LiString *t)
{
   int k;
   LiString *str,*p=s->next,*p1=t->next,*q,*r;
   str=(LiString *)malloc(sizeof(LiString));
   //参数非法时的处理
   r=str;
   //①将s的前i-1个节点复制到str
   //②让p沿next跳j个节点
   //③将t的所有节点复制到str
   //④将p及其后的节点复制到str
   r->next=NULL;
   return str;
}
```

```
①for (k=0; k<i-1; k++)
   {
      q=(LiString *)malloc(sizeof(LiString));
      q->data=p->data;
      q->next=NULL;
      r->next=q;
      r=q;
      p=p->next;
   }
```

```
②for (k=0; k<j; k++)
      p=p->next;
```

```
③ while (p1!=NULL)
   {
      q=(LiString *)malloc(sizeof(LiString));
      q->data=p1->data;
      q->next=NULL;
      r->next=q;
      r=q;
      p1=p1->next;
   }
```

```
④while (p!=NULL)
   {
      q=(LiString *)malloc(sizeof(LiString));
      q->data=p->data;
      q->next=NULL;
      r->next=q;
      r=q;
      p=p->next;
   }
```



输出串：DispStr(s)

显示串中的字符

```
void DispStr(LiString *s)
{
    LiString *p=s->next;
    while (p!=NULL)
    {
        printf("%c",p->data);
        p=p->next;
    }
    printf("\n");
}
```