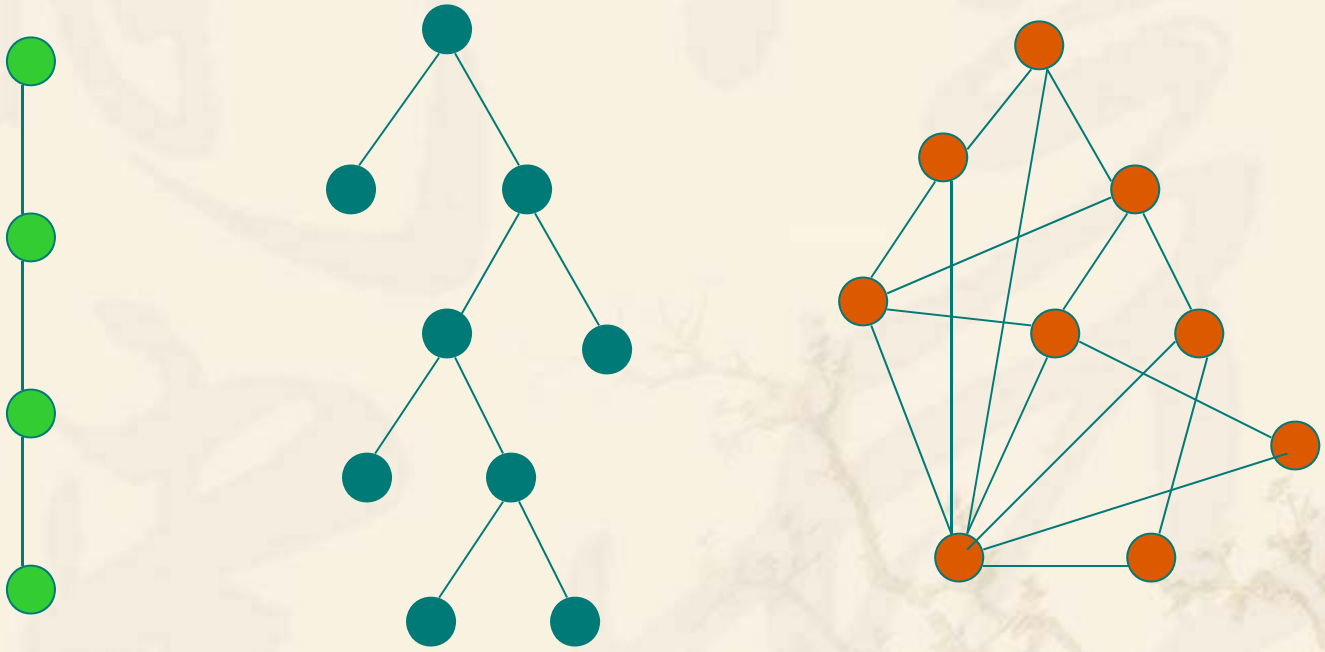
数据结构

华中科技大学

计算机学院



第四章，字符串/串(string)

4.1 串的定义与操作

1.术语

(1)串：由零个或多个字符组成的有限序列。

- n个字符C1,C2,...,Cn组成的串记作：

  s=’$C_1C_2...C_n$’　　n>=0

其中:s为串名，　$C_1C_2...C_n$为串值　　n为串长

例 PASCA语言:

　　s1=’data1234’　　s2=’123*abc’

C语言:

　　s1=″data1234″　　s2=″123*abc″

　　’A’为字符　　″A″为字符串

　　%c为字符格式　　%s为字符串格式

(2)空串：不含字符的串/长度为零的串。

PASCA语言：　s=’’

C语言：　　s=″″

(3)空格串：仅含空格字符’　’的串。

例　s1=’Φ’　　s2=’ΦΦ’

　　s1=’　’　　s2=’　‘

(4)子串：串s中任意个连续的字符组成的子序列称为串s的子串。

主串--- 包含某个子串的串。

例　st=″ABC123A123CD″

　　s1=″ABC″　　s3=″123A″　　s4=″ABCA″

　　s2=″ABC12″　　s5=″ABCE″　　s6=″321CBA″

2. 串变量、字符变量的定义与使用

例1 串变量

```
char st[]="abc\'*123";
gets(st);  scanf("%s",st);
strcpy(st,"data");
puts(st);  printf( “st=%s\n” ,st);
```

例2 字符变量

```
char ch='A';
ch=' B' ;  ch=getchar();
scanf("%c",&ch);  printf("ch=%c\n",ch);
```

3.串的基本操作与串函数

(1)strcpy(t,s)---s的串值复制到t中。

执行：strcpy(t,"data")；　有：t="data"

(2)strlen(s)----求s的长度

strlen("data*123")=8　strlen("")=0

(3)strcat(s1,s2)----s2的值接到s1的后面。

设 s1="data"，　s2="123"

执行：strcat(s1,s2);

有：　s1="data123"，　s2="123"

(4)strcmp(s1,s2)---比较s1和s2的大小

若s1<s2,返回负整数　　如："ABC"<"abc"

若s1=s2,返回0如：　　如："abc"="abc"

若s1>s2,返回正整数　　如："ABCD">"ABC"

(5) strstr(s1,s2)----若s2是s1的子串，则指向s2在s1中第1次出现时的第1个字符的位置；否则返回NULL。

设 s1="ABE123*DE123bcd"

s2="E123"

有 strstr(s1,s2)=3

s1="ABE123*DE123"

(6)Replace(s,t,v)----置换

用v代替主串s中出现的所有与t相等的不重叠的子串

设 s="abc123abc*ABC", t="abc", v="OK"

执行：Replace(s,t,v);

有： s="OK123OK*ABC", t="abc", v="OK"

设 A="abcaaaaaABC", B="aa", C="aaOK"

执行：Replace(A,B,C);

有： A="abcaaOKaaOKaABC", B="aa", C="aaOK“

(7) StrInsert(s,i,t)----将t插入s中第i个字符之前。

设 s="ABC123"

执行: StrInsert(s,4,"**");

有: s="ABC**123"

(8)用Replace(s,t,v)实现删除

设　　s="ABC//123"

执行：Replace(s,"//","")

有：　s="ABC123"

(9)用Replace(s,t,v)实现插入

设　　s="ABC123"

执行：Replace(s,"123","**123")

有：　s="ABC**123"

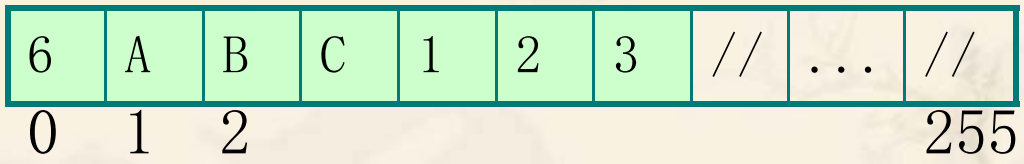
4.2 串的存储表示和实现

4.2.1 串的定长顺序存储表示

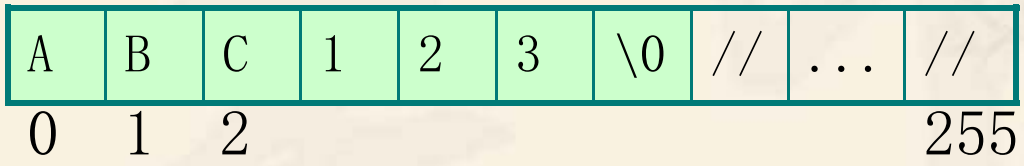
给每个定义的串分配一个固定长度的存储区

```
#define   MAXSTRLEN    255    //用户可在255以内定义最大长度
typedef unsigned char SString[MAXSTRLEN+1]; //0号单元存放
                                            //串的长度
```

PASCAL: 下标为0的分量存放串的实际长度

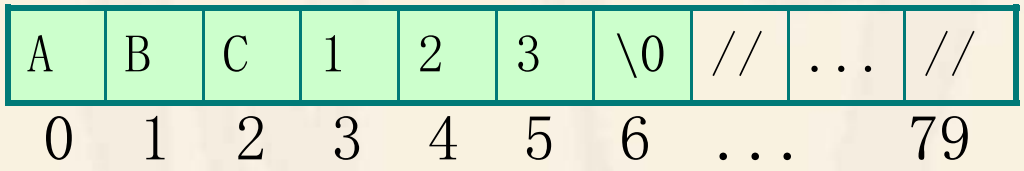


C: 在串值后加串结束标记‘\0’,串长为隐含值

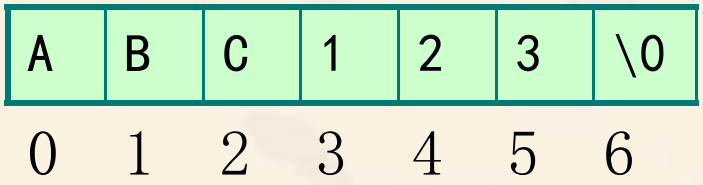


1. **顺序存储**----用一维字符数组表示一个串的值。

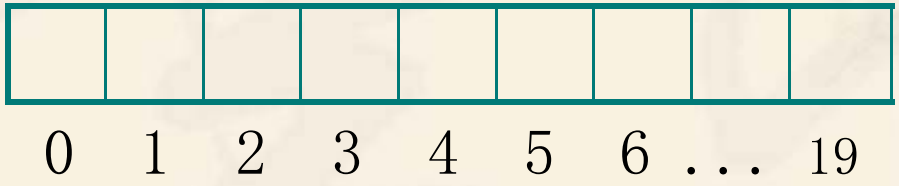
例1 char st1[80]=”ABC123”;



例2 char st2[]=”ABC123”;



例3 char st[20];



2. 串运算实现举例

例 联接运算：给定串s1,s2,将s1,s2联接为串t,记作：

Concat(t,s1,s2)

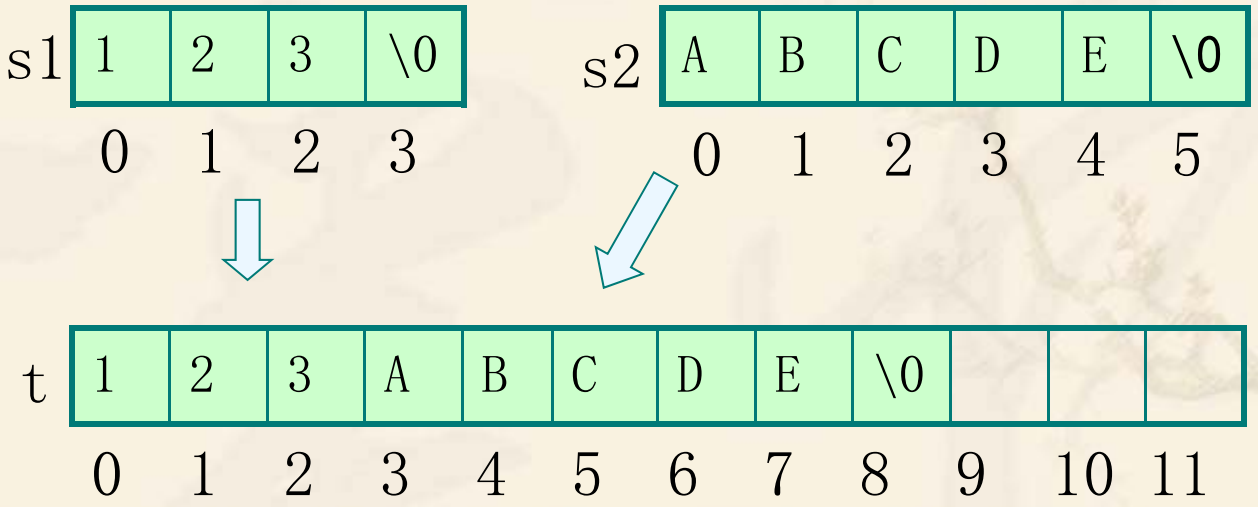
设 s1=”123”, s2=”ABCDE”

执行： Concat(t,s1,s2)；

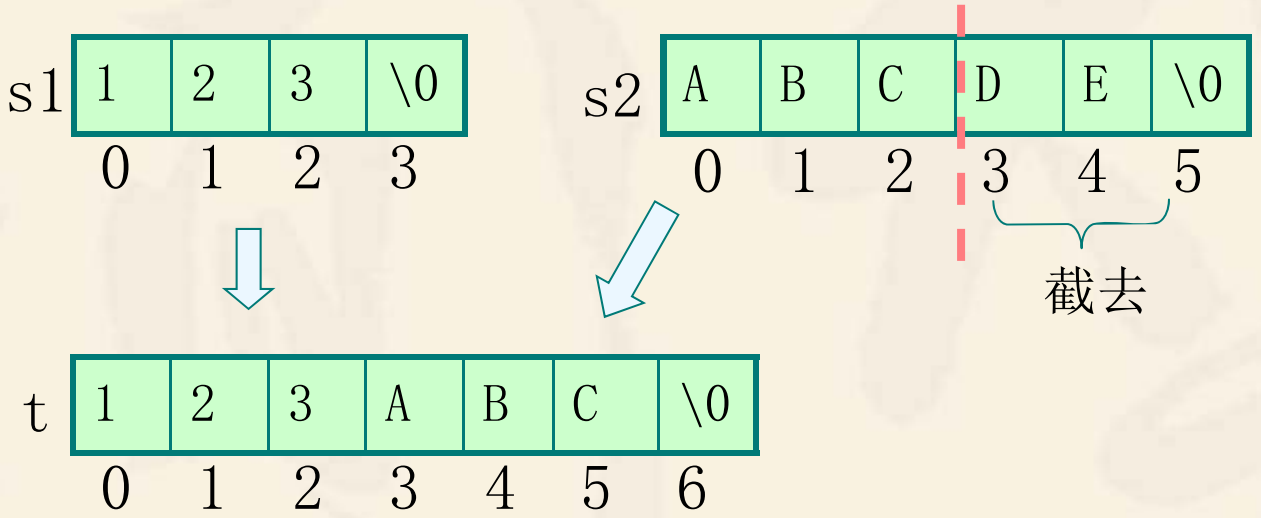
有： t=”123ABCDE”

实现Concat(t,s1,s2)的方法：

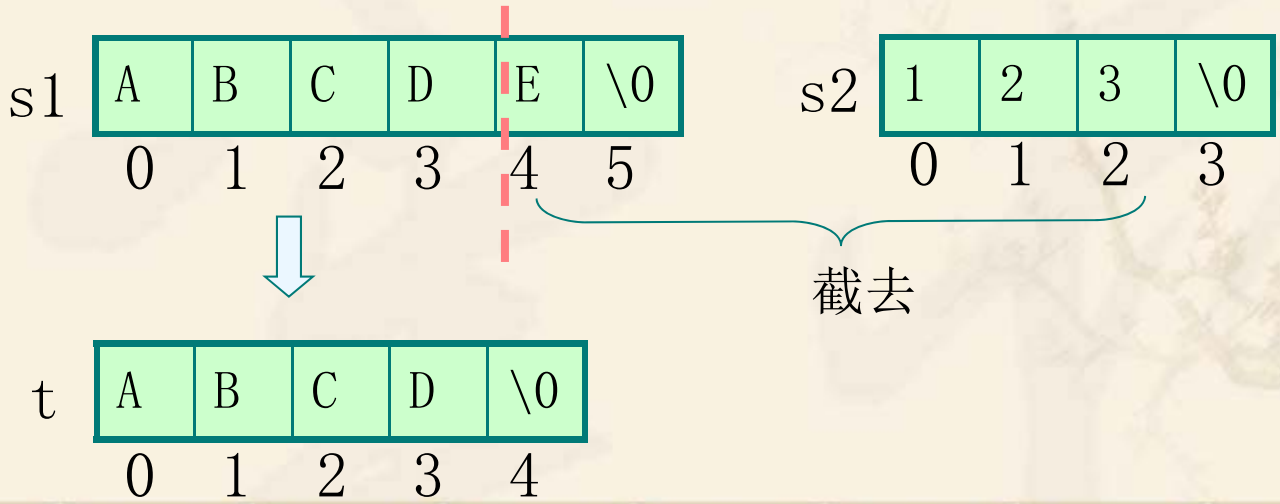
(1)s1的长度+s2的长度≤t的最大长度：



(2)s1的长度≤t的最大长度≤s1的长度+s2的长度：

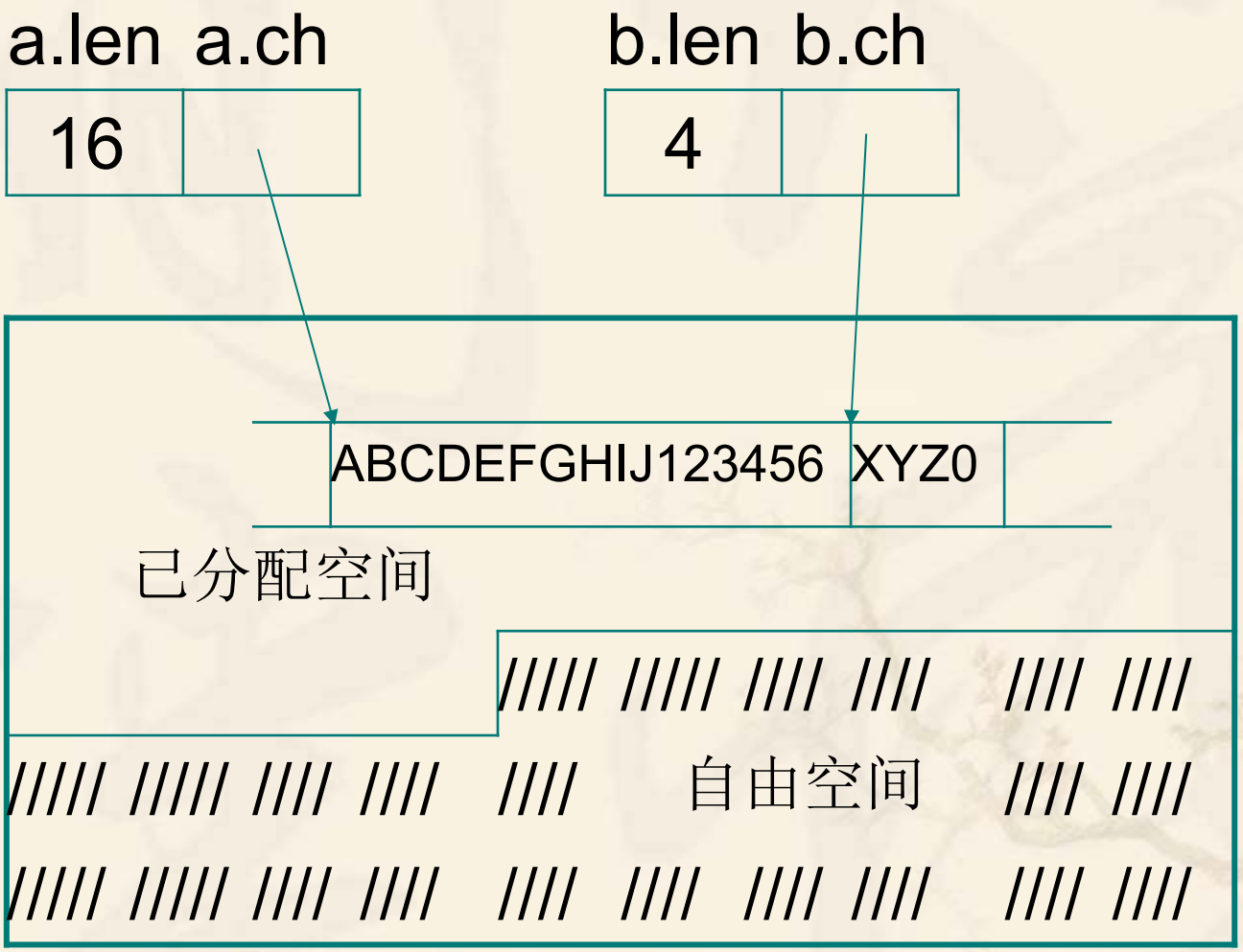


(3) t的最大长度<s1的长度：



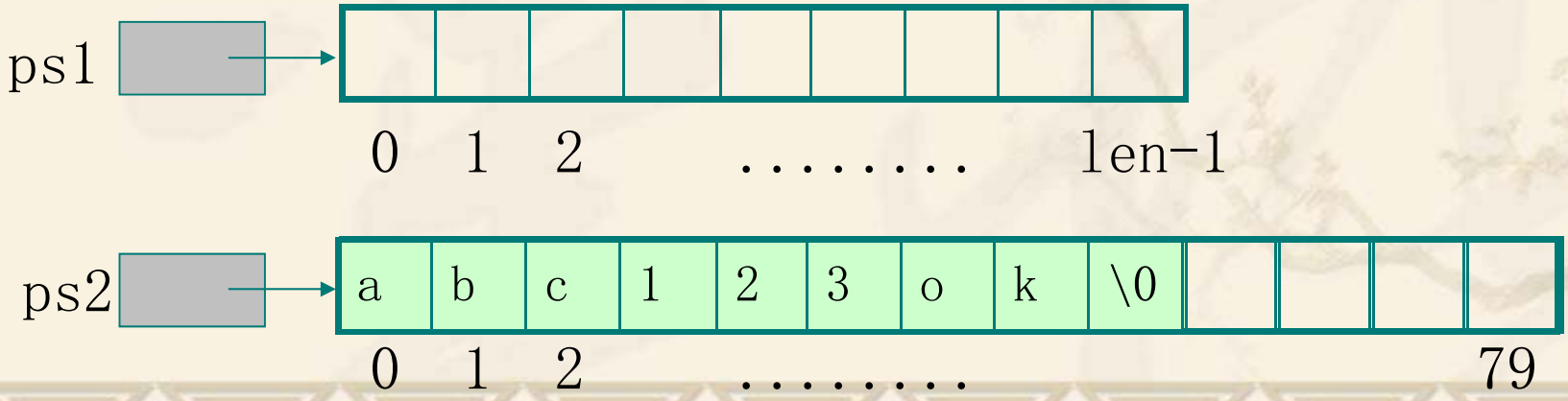
4.2.2 串的堆分配存储表示

提供一个足够大的连续存储空间，存放字符串的值。



例1：C语言中利用动态分配使用系统堆。

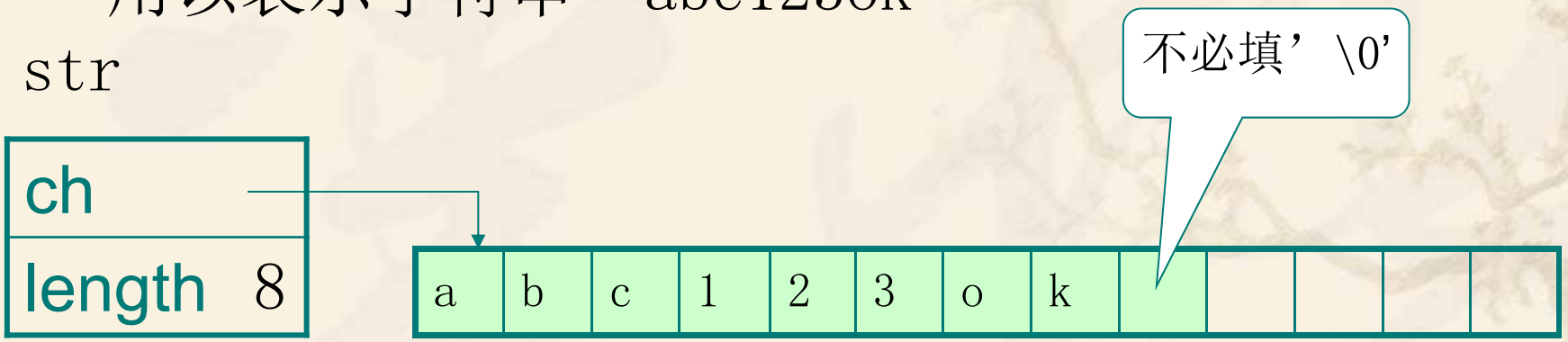
```
{ char *ps1,*ps2; int len;
  scanf("%d",&len);                    //输入长度值
  ps1=(char *)malloc(len);             //ps1指向分配的存储空间
  gets(ps1);  puts(ps1);               //输入一个串，再输出
  ps2=(char *)malloc(80);              //ps2指向分配的存储空间
  strcpy(ps2,"abc123ok") ;             //赋值，再输出
  puts(ps2);
  free(ps1); free(ps2);                //释放存储空间
 }
```



堆存储结构描述,定义将串长作为存储结构的一部分：

```
typedef  struct {
    char   *ch;      //若是非空串，则按串长
                     //分配存储区，否则ch为NULL
    int    length; //串长度
    }  HString;
HString  str;
```

用以表示字符串“abc123ok”



例2：字符串赋值操作

```
int StrAssign( HString *T,  char *chars)
{  char *c;  int i;
   if (T->ch) free(T->ch);              /*释放T原有空间*/
   for(i=0,c=chars; *c; i++,++c);       /*求chars的串长i*/
   if (!i)
      {T->ch=NULL;T->length=0;}         /*当chars为空串时*/
   else {
         if (!(T->ch=(char *) malloc(i*sizeof(char))))
                  return OVERFLOW;
         T->length=i;
         for(i; i>0; i--)               /*复制chars串值到串T*/
              T->ch[i-1]=chars[i-1];
         }
   return OK;
 }
```

例3：输出字符串

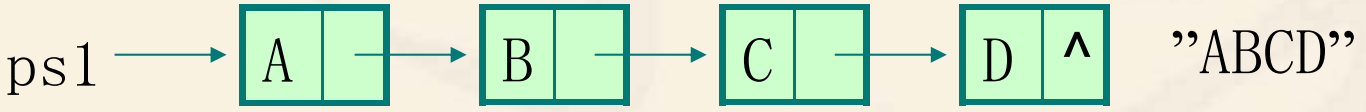
```
void StrPrint(HString T)
{
 int i;
 for(i=0;i<T.length;i++)
     putchar(T.ch[i]);
 }

void  main(void)
{
  HString str;
  StrAssign(&str,”abcd123”);
  StrPrint(str);
 }
```

4.2.3 串的单链表表示

例1 一个结点只放1个字符

```
struct node1
 { char data;              //为一个字符
   struct node1 *next;     //为指针
  }*ps1;
```

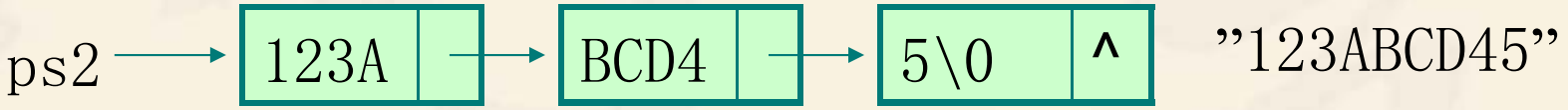


存储密度=串值所占存储位/实际分配存储位

存储密度为 0.33(4/((1+2)*4)=1/3)

例2 一个结点放4个字符

```
struct node4
{ char data[4];          //为4个字符的串
  struct node4 *next;    //为指针
 }*ps2;
```



存储密度为0.5(9/((4+2)*3)=9/18=1/2)

课后作业

《数据结构题集》

4.1