**第四章 串**

**串（又称字符串）是一种特殊的线性表，其数据元素由字符构成。**

**4.1 串的逻辑结构**

**1．串的定义**

**串是由0个或n个字符组成的序列。记为：**

**s=“”**

**其中：s是串名，双引号是定界符，引起的内容称为串值，n是串长，当n=0时，称为空串，记为Φ。**

**2．关于串的几个术语**

**子串与主串：串中任意连续字符组成的子序列称为该串的子串，包含子串的串称为主串。**

**如：a=“BeiJing”,b=“Bei”,c=“Jing”**

**b和c是a的子串，a是b和c的主串**

**子串的位置：子串的第一个字符在主串中的序号称为子串的位置。**

**如b和c在a中的位置分别是1和4。**

**串相等（要求严格）：是指两个串的长度和对应字符完全相等。**

**如:a=“BeiJing”，b=“beijing”，c=“Beijing”**

**这3个字符串都不相等。**

**4.2 串的基本操作**

**对串可以进行输入、输出、求串长、拷贝、连接、比较和求字串的位置等。**

**在C语言中，有一系列的字符串处理库函数。**

**#include<stdio.h> Turbo C**

**#include<string.h> 标准C**

**设char s1[20],s2[15];**

1. **字符串输入函数：gets(字符数组名)；**

**空格可以作为内容输入，换行作为结束标志。**

**例：gets(s1);**

**输入：I am a student. ↙**

1. **字符串输出函数：puts(字符数组名)；**

**‘\0’变为换行。**

**例：puts(s1);**

**输出内容：I am a student.**

**(3)求字符串的长度（不含\0）：strlen(字符数组名或字符串)；**

**例：printf(“%d”,strlen(s1));**

**输出：15**

**printf(“%d”,strlen(“BUAA”));**

**输出：4**

**P4-1.c 字符串的gets（输入）,puts（输出）和strlen（长度）的使用**

**#include<stdio.h>**

**void main()**

**{**

**char s1[20];**

**printf("输入字符串s1： ");**

**gets(s1);**

**printf("字符串： ");puts(s1);**

**printf("s1的长度是 %d\n",strlen(s1));**

**printf("字符串\"BUAA\"的长度是%d\n",**

**strlen("BUAA"));**

**}**

**(4)字符串的拷贝：strcpy(字符数组名，字符数组名或字符串)**

**例：strcpy(s2,“I am a student.”)；**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s2:** | **I** |  | **a** | **m** |  | **a** |  | **s** | **t** | **u** | **d** | **e** | **n** | **t** | **.** | **\0** |

**strcpy(s1,s2);**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s1:** | **I** |  | **a** | **m** |  | **a** |  | **s** | **t** | **u** | **d** | **e** | **n** | **t** | **.** | **\0** |  |  |  |  |

**(5)字符串的连接：strcat(较大字符数组名，字符数组名或字符串)**

**前一串的\0被后串的第一个字符覆盖。**

**例：strcpy(s2,“BEIJING”);**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s1:** | **C** | **H** | **I** | **N** | **A** | **\0** |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s2:** | **B** | **E** | **I** | **J** | **I** | **N** | **G** | **\0** |  |  |

**strcat(s1,s2);**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s1:** | **C** | **H** | **I** | **N** | **A** | **B** | **E** | **I** | **J** | **I** | **N** | **G** | **\0** |  |  |  |

**(6) 字符串的比较：strcmp(字符数组名或字符串1，字符数组名或字符串2)**

**正数 当串1 > 串2**

**🡪 函数的返回值= 0 当串1 = 串2 **

**负数 当串1 < 串2**

**说明：两个字符串若长度不同，短者用\0补齐，再自左至右逐个字符进行比较，直至分出大小或相等。**

1. **串s1和串s2完全相等的情况**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s1:** | **B** | **U** | **A** | **A** | **\0** |  |  |  |  |
| **s2:** | **B** | **U** | **A** | **A** | **\0** |  |  |  |  |

**printf(“%d”,strcmp(s1,s2))；**

**或printf(“%d”,strcmp(“BUAA”,“BUAA”))；**

**输出: 0**

**（2）串s1 >串s2的情况**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s1:** | **C** | **H** | **I** | **N** | **A** | **\0** | **\0** | **\0** |  |
| **s2:** | **B** | **E** | **I** | **J** | **I** | **N** | **G** | **\0** |  |

**printf(“%d”,strcmp(s1,s2))；**

**输出: 1（‘C’-‘B’的编码之差）**

**注意：比较时两串左部对齐，短者用\0补齐**

**（3）串s1<串s2的情况**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s1:** | **w** | **h** | **i** | **l** | **e** | **\0** |  |  |  |
| **s2:** | **w** | **h** | **i** | **t** | **e** | **\0** |  |  |  |

**printf(“%d”,strcmp(s1,s2))；**

**输出: -2（‘l’-‘t’ 的编码之差）**

**P4-2.c 字符串的strcpy（拷贝）,strcat（连接）和strcmp（比较）的使用**

**#include<stdio.h>**

**void main()**

**{**

**char s1[20],s2[20],s3[20]="BUAA";**

**strcpy(s1,"China");**

**printf("将字符串\"China\"拷贝到s1,**

**长度=%d\n",strlen(s1));**

**printf("字符串s1:");puts(s1);**

**strcpy(s2,"Beijing");**

**printf("\n将字符串\"Beijing\"拷贝到s2,**

**长度=%d\n",strlen(s2));**

**printf("字符串s2:");puts(s2);**

**strcat(s1,s2);printf("\n连接s1+s2->s1,**

**长度=%d\n",strlen(s1));**

**printf("字符串s1:");puts(s1);**

**printf("\n字符串s3:");puts(s3);**

**printf("长度=%d\n",strlen(s3));**

**strcat(s1,s3);printf("\n连接s1+s3->s1,**

**长度=%d\n",strlen(s1));**

**printf("字符串s1:");puts(s1);**

**printf("\n\"while\" and \"white\":%d\n\n",**

**strcmp("while","white"));**

**printf("\n\"BUAA and \"BUAA\":%d\n\n",**

**strcmp(s3,"BUAA"));**

**printf("\n\"China\" and \"Beijing\":%d\n",**

**strcmp("China","Beijing"));**

**}**

**4.3 串的存储结构**

**串的存储分为两种：顺序存储和链式存储**

**4.3.1串的顺序存储**

**串的顺序存储结构称为顺序串，类似于顺序表，按存储空间分配的方式不同，分为静态顺序串和动态顺序串。**

**1.静态存储结构的顺序串**

**(1)串长隐式方式**

**事先定义一个字符型数组，在串的尾部加一个特**

**殊标志‘\0’（C语言采用的这种方式），该标志不计入长度。**

**#define MAXSIZE 100**

**char s[MAXSIZE];**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **s[0]** | **s[1]** | **s[2]** | **s[3]** | **s[4]** | **…** | **s[MAXSIZE-1]** |
| **a** | **b** | **c** | **\0** |  |  |  |

**（2）串长显式方式**

**不设串结束标志，增设一个整型变量计串长。**

**#define MAXSIZE 100**

**typedef struct**

**{**

**char ch[MAXSIZE];**

**int length;**

**}SeqString;**

**SeqString s;**

**其中的串长=s.length=4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **s[0]** | **s[1]** | **s[2]** | **s[3]** | **s[4]** | **…** | **s[MAXSIZE-1]** |
| **a** | **b** | **c** | **d** |  |  |  |

**(3)对于静态顺序串的说明**

**·与顺序表类似，插入和删除等不方便；**

**·串长受MAXSIZE规定的限制，超长时通常采用“截尾”处理。**

**2. 动态存储结构的顺序串**

**存储空间不受限制。**

**typedef char \*String;**

**或**

**typedef struct**

**{**

**char \*str;**

**int length;**

**}String;**

**4.2.2串的链式存储结构**

**简称链串。如每个结点只放一个字符的链串,称为结点大小为1的链串**

**typedef struct node**

**{**

**char ch;**

**struct node \*next;**

**}LinkString;**

**LinkString \*s;**

**例如：s=“abcd”**

a

b

c

d

**s🡪**

**串值的存储密度=串值的存储/实际分配的存储**

**密度越小越合算，反之则不然。为解决密度过低，一个结点存储多个字符，长度结点不一定是结点的整倍数，最后一个串的结点用特殊字符填充。**

**如结点大小为3的链串**

**#define NODESIZE 4**

**typedef struct node**

**{**

**char ch[NODESIZE];**

**struct node \*next;**

**}LinkString;**

**LinkString \*s;**

**s=“abcdefghij”**

abc

def

ghi

j\0\0

**s🡪**

**这种结点大于1的链串，虽提高了密度，但在进行插入和删除等操作时，会引起大量的字符移动，不方便。**

**4.3串的模式匹配**

**子串定位（模式匹配）法：确定子串在串中起始位置。**

**主串s称为目标串，子串p称为模式串。**

**简单的模式匹配算法：从模式串（子串）p中的第一个字符开始与目标串（主串）s中的字符进行比较，若相等，接着比下去，否则从目标串s的当前位置的下一个字符开始下一趟的模式匹配。**

**例如：目标串s=“ababcdbbcdacba”**

**模式串p=“bcdac”//到第8趟匹配成功**

**模式串q=“bcac”//自始至终不能匹配**

|  |  |
| --- | --- |
| **第趟(k)** | **1 2 3 4 5 6 7 8 9 ……** |
| **目标串s(i)**  **模式串p(j)**  **模式串q(j)** | **a b a b c d b b c d a c b a**  **b c d a c**  **b c a c** |

**第1趟：k=1**

**i=1**

**目标串s=a b a b c d b b c d a c b a**

**≠**

**模式串p=b c d a c**

**j=1**

**第2趟：k=2**

**i=2 i=3**

**目标串s=a b a b c d b b c d a c b a**

**= ≠**

**模式串p= b c d a c**

**j=1 j=2**

**第3趟：k=3**

**i=3**

**目标串s=a b a b c d b b c d a c b a**

**≠**

**模式串p= b c d a c**

**j=1**

**第4趟：k=4**

**i=4 i=5 i=6 i=7**

**目标串s=a b a b c d b b c d a c b a**

**= = = ≠**

**模式串p= b c d a c**

**j=4**

**第5趟：k=5**

**i=5**

**目标串s=a b a b c d b b c d a c b a**

**≠**

**模式串p= b c d a c**

**j=1**

**第6趟：k=6**

**i=6**

**目标串s=a b a b c d b b c d a c b a**

**≠**

**模式串p= b c d a c**

**j=1**

**第7趟：k=7**

**i=8**

**目标串s=a b a b c d b b c d a c b a**

**= ≠**

**模式串p= b c d a c**

**j=2**

**第8趟：k=8**

**i=13**

**目标串s=a b a b c d b b c d a c b a**

**= = = = =**

**模式串p= b c d a c**

**j=6**

**例p4-3.c 子串定位的简单模式匹配法**

**#define MAXSTRSIZE 100**

**#include<stdio.h>**

**typedef struct**

**{**

**char str[MAXSTRSIZE];**

**int length;**

**}SeqString;**

**SeqString s={" ababcdbbcdacba",14},**

**p={" bcdac",5};**

**//注意字符串是从s[1]和p[1]开始**

**int SimpleMatching(SeqString \*s,SeqString \*p)**

**{**

**int i,j,k,m=s🡪length,n=p🡪length;**

**for(k=1;k<=m-n+1;k++)**

**{**

**i=k;j=1;**

**while(j<=n&&s🡪str[i]==p🡪str[j])**

**{**

**i++;j++;**

**}**

**if(j>n)return k;//第k趟匹配成功**

**}**

**return -1;//匹配不成功**

**}**

**void main()**

**{ int pos;**

**pos=SimpleMatching(&s,&p);**

**if(pos!=-1)printf("子串的位置是：%d\n",pos);**

**else printf("匹配失败！\n");**

**}**