第4章

串

本章学习导读

在计算机的各方面应用中，非数值处理问题的应用越来越多。如在汇编程序和编译程序中，源程序和目标程序都是作为一种字符串数据进行处理的。在事务处理系统中，用户的姓名和地址及货物的名称、规格等也是字符串数据。

字符串一般简称为串，可以将它看作是一种特殊的线性表，这种线性表的数据元素的类型总是字符型的，字符串的数据对象约束为字符集。在一般线性表的基本操作中，大多以“单个元素”作为操作对象，而在串中，则是以“串的整体”或一部分作为操作对象。因此，一般线性表和串的操作有很大的不同。本章主要讨论串的基本概念、存储结构和一些基本的串处理操作。

4.1 串的基本概念

4.1.1 串的定义

串(或字符串)(String)是由零个或多个字符组成的有限序列。一般记作：

$$s=“c_0 c_1 c_2 \cdots c_{n-1}” \quad (n\geqslant 0)$$

其中：

(1)s为串名，用双引号括起来的字符序列是串的值；c_i($0\leqslant i\leqslant n-1$)可以是字母、数字或其他字符；

(2)双引号为串值的定界符，不是串的一部分；

(3)字符串字符的数目n称为串的长度。零个字符的串称为空串，通常以两个相邻的双引号来表示空串(Null string)，如：s=“”，它的长度为零；仅由空格组成的串称为空格串，如：s=“␣”；若串中含有空格，在计算串长时，空格应计入串的长度中，如：s=“I’m a student”的长度为13。

请读者注意，在C语言中，用单引号引起来的单个字符与单个字符的串是不同的，如s1=‘a’与s2=“a”两者是不同的，s1表示字符，而s2表示字符串。

4.1.2　主串和子串

一个串的任意个连续的字符组成的子序列称为该串的子串，包含该子串的串称为主串。称一个字符在串序列中的序号为该字符在串中的位置，子串在主串中的位置是以子串的第一个字符在主串中的位置来表示的。当一个字符在串中多次出现时，以该字符第一次在主串中出现的位置为该字符在串中的位置。

例如：s1、s2、s3 为如下的三个串：s1＝“I’m a student”，s3＝“student”，s2＝“teacher”。

则它们的长度分别为 13、7、7。串 s3 是 s1 的子串，子串 s3 在 s1 中的位置为 7，也可以说 s1 是 s3 的主串。串 s2 不是 s1 的子串，串 s2 和 s3 不相等。

4.2　串的存储结构

对串的存储方式取决于我们对串所进行的运算。如果在程序设计语言中，串的运算只是作为输入或输出的常量出现，则此时只需存储该串的字符序列，这就是串值的存储。此外，一个字符序列还可赋给一个串变量，操作运算时通过串变量名访问串值。实现串名到串值的访问，在 C 语言中可以有两种方式：

一、是可以将串定义为字符型数组，数组名就是串名，串的存储空间分配在编译时完成，程序运行时不能更改，这种方式为串的静态存储结构；

二、是定义字符指针变量，存储串值的首地址，通过字符指针变量名访问串值，串的存储空间分配是在程序运行时动态分配的，这种方式称为串的动态存储结构。

4.2.1　串值的存储

我们称串是一种特殊的线性表，因此串的存储结构表示也有两种方法：静态存储采用顺序存储结构，动态存储采用的是链式存储和堆存储结构。

1. 串的静态存储结构

类似于线性表的顺序存储结构，用一组地址连续的存储单元存储串值的字符序列。由于一个字符只占 1 个字节，而现在大多数计算机的存储器地址是采用的字编址，一个字（即一个存储单元）占多个字节，因此顺序存储结构方式有两种：

(1)紧缩格式：即一个字节存储一个字符。这种存储方式可以在一个存储单元中存放多个字符，充分地利用了存储空间。但在串的操作运算时，若要分离某一部分字符时，则变得非常麻烦。

d	a	t	a
␣	S	t	r
u	c	t	u
r	e	\0	

图 4-1　串值的紧缩格式存储

图 4-1 所示是以 4 个字节为一个存储单元的存储结构，每个存储单元可以存放 4 个字符。

对于给定的串 s="data ␣ structure",在 C 语言中采用字符'\0'作串值的结束符。串 s 的串值连同结束符的长度共 15,只需 4 个存储单元。

(2)非紧缩格式:这种方式是以一个存储单元为单位,每个存储单元仅存放一个字符。这种存储方式的空间利用率较低,如一个存储单元有 4 个字节,则空间利用率仅为 25%。但这种存储方式中不需要分离字符,因而程序处理字符的速度高。图 4-2 即为这种结构的示意图。

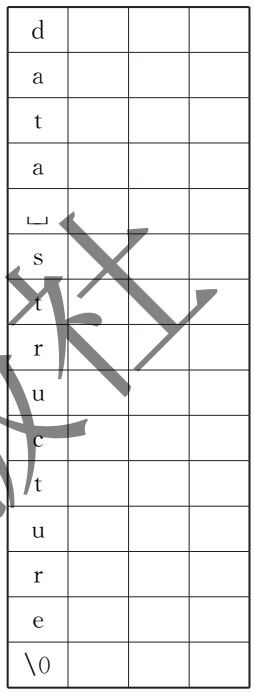

图 4-2 串值的非紧缩格式存储

用字符数组存放字符串时,其结构用 C 语言定义如下:

```
#define MAXNUM <允许的最大的字符数>

typedef struct
{
  char str[MAXNUM];
  int length; /* 串长度 */
} stringtype; /* 串类型定义 */
```

由上述讨论可知,串的顺序存储结构有两大不足之处。一是需事先预定义串的最大长度,这在程序运行前是很难估计的,二是由于定义了串的最大长度,使得串的某些操作受限,如串的联接运算等。

2.串的动态存储结构

我们知道,串的各种运算与串的存储结构有着很大的关系,在随机存取子串时,顺序存储方式操作起来比较方便,而对串进行插入、删除等操作时,顺序存储方式使操作变得很复杂。因此,有必要采用串的动态存储方式。

串的动态存储方式采用链式存储结构和堆存储结构两种形式:

(1)链式存储结构

串的链式存储结构中每个结点包含字符域和结点链接指针域,字符域用于存放字符,指针域用于存放指向下一个结点的指针,因此,串可用单链表表示。

用链表存放字符串时,其结构用 C 语言定义如下:

```
typedef struct node{
char str;
struct node  *next;
} slstrtype;
```

用单链表存放串,每个结点仅存储一个字符,如图 4-3 所示,因此,每个结点的指针域所占空间比字符域所占空间要大得多。为了提高空间的利用率,我们可以使每个结点存放多个字符,称为块链结构,如图 4-4 所示每个结点存放 4 个字符。

图 4-3 串的链式存储结点大小为 1 的链表

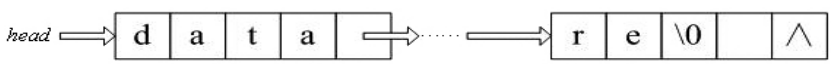

图 4-4 串的链式存储结点大小为 4 的链表

用块链存放字符串时，其结构用 C 语言定义如下：

```
typedef struct node{
char str[4];
struct node   * next;
} slstrtype;
```

(2)堆存储结构。

堆存储结构的特点是仍以一组空间足够大的、地址连续的存储单元存放串值字符序列，但字符串的存储空间是在程序执行过程中动态分配的。每当产生一个新串时，系统就从剩余空间的起始处为串值分配一个长度和串值长度相等的存储空间。

在 C 语言中，存在一个称为“堆”的自由空间，由动态分配函数 malloc()分配一块实际串长所需的存储空间，如果分配成功，则返回一个指向这段空间的起始地址的指针，作为串的基址。由 free()释放串不再需要的空间。

用堆存放字符串时，其结构用 C 语言定义如下：

```
typedef struct{
  char * str;
  int length;
} HSstrtype;
```

4.2.2　串名的存储映像

串名的存储映像就是建立了串名和串值之间的对应关系的一个符号表。在这个表中的项目可以依据实际需要来设置，以能方便地存取串值为原则。

如：

s1＝“data”

s2＝“structure”

假若一个单元仅存放 1 个字符，则上面两个串的串值顺序存储如图 4-5 所示。

若符号表中每行包含有串名、串值的始地址、尾地址，则如图 4-6(a)所示，也可以不设尾地址，而设置串名、串值的始地址和串的长度值。则如图 4-6(b)所示。

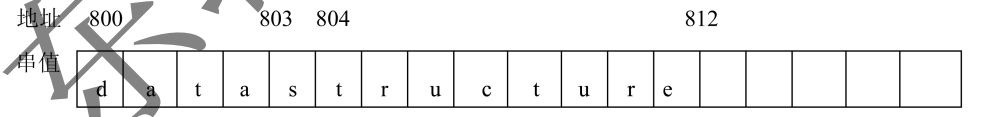

图 4-5　串 s1 和 s2 的存储状态

串名	始地址	尾地址
S1	800	803
S2	804	812

(a)

串名	始地址	串长
S1	800	4
S2	804	9

(b)

图 4-6　符号表示例

对于链式存储串值的方式，如果要建立串变量的符号表，则只需要存入一个链表的表头指针即可。

4.3 串的基本运算及其实现

串的基本运算有赋值、联接、求串长、求子串、求子串在主串中出现的位置、判断两个串是否相等、删除子串等。在本节中,我们尽可能以 C 语言的库函数表示其中的一些运算,若没有库函数,则用自定义函数说明。

4.3.1 串的基本运算

(1)strcpy(str1,str2) 字符串拷贝(赋值):把 str2 指向的字符串拷贝到 str1 中,返回 str1。库函数和形参说明如下:

```
char * strcpy(char * str1,char * str2)
```

(2)strcat(str1,str2) 字符串的联接:把字符串 str2 接到 str1 后面,str1 最后的结尾符'\0'被取消。返回 str1。库函数和形参说明如下:

```
char * strcat(char * str1,char * str2)
```

(3)strlen(str) 求字符串的长度:统计字符串 str 中字符的个数(不包括'\0'),返回字符的个数,若 str 为空串,则返回值为 0。库函数和形参说明如下:

```
unsigned int strlen(char * str)
```

(4)strstr(str1,str2) 子串的查询:找出子串 str2 在主串 str1 第一次出现的位置(不包括子串 str2 的结尾符),返回该位置的指针,若找不到,返回空指针 NULL。库函数和形参说明如下:

```
char * strstr(char * str1,char * str2)
```

(5)strcmp(str1,str2) 字符串的比较:比较两个字符串 str1、str2。若 str1<str2,则返回负数;若 str1>str2,则返回正数;若 str1=str2,则返回 0。库函数和形参说明如下:

```
int strcmp (char * str1,char * str2)
```

(6)substr(str1,str2,m,n) 求子串:在字符串 str1 中,从第 m 个字符开始,取 n 个长度的子串 str2。若 m>strlen(str)或 n<0,则返回空值 NULL。自定义函数和形参说明如下:

```
int substr(char * str1,char * str2,int m,int n)
```

(7)delstr(str,m,n) 字符串的删除:在字符串 str 中,删除从第 m 个字符开始的 n 个长度的子串。

(8)insstr(str1,m,str2) 字符串的插入:在字符串 str1 第 m 个位置之前开始,插入字符串 str2。返回 str1。

对字符串的置换可以通过求串长,删除子串,字符串的联接等基本运算来实现。

4.3.2 串的基本运算及其实现

本小节中,我们将讨论串值在静态存储方式和动态存储方式下,其运算如何实现。

如前所述,串的存储可以是静态的,也可以是动态的。静态存储在程序编译时就分配了存储空间,而动态存储只能在程序执行时才分配存储空间。不论在哪种方式下,都能实现串的基本运算。

1.在静态存储结构方式下进行存储

C 语言中用字符数组存储字符串时,结构定义如下:

```
#define MAXNUM 80
typedef struct {
  char str[MAXNUM];
  int length; /* 串长度 */
} stringtype; /* 串类型定义 */
```

求将字符串存储到字符数组中,串的长度可以通过函数 strlen()计算得到。算法如下:

【算法 4.1】静态存储方式

```
void main()
{
    stringtype s1;
    gets(s1.str1);                          //字符串的存储
    s1.length=strlen(s1.str1);              //计算字符串的长度
    printf("\n%s,%d",s1.str1,s1.length);
}
```

上述算法中使用数组存放字符串,串长用显式方式给出。

2.在动态存储结构方式下进行存储

(1)在链式存储结构方式下。

假设链表中每个结点仅存放一个字符,则单链表定义如下:

```
typedef struct node
{
    char str;
    struct node *next;
};
```

创建链表的算法如下:

【算法 4.2 】链式存储方式

```
struct node *creat(void)
{
    struct node *head;
    struct student *p1,*p2;
    int n=0;
    p1=p2=(struct node *)malloc(LEN);
    scanf("%c",&p1->str);
    head=NULL;
    while(p1->str!='0')
    {
        n=n+1;
        if(n==1)
            head=p1;
        else
```

```
            p2->next=p1;
        p2=p1;
        p1=(struct node *)malloc(LEN);
        scanf("%c",&p1->str);
    }
    p2->next=NULL;
    return(head);
    }
```

上述算法中,每个结点只存放一个字符,字符串的存储空间浪费很大,我们可以使用块链结构存储串,有兴趣的读者可以设计该算法。

(2)在堆存储结构方式下。

堆存储结构用 C 语言定义为:

```
typedef struct Hsstrtype
{
char * str;
int length;
};
```

在创建存储空间时需要创建两个存储空间,因为字符串并不是存储在结构体中,所以需要另外分配一个空间。

【算法 4.3】链式存储方式

```
#define MAXNUM 800
typedef struct HSstrtype
{
char * str;
int length;
};
void main()
{
    struct HSstrtype * s1;
    printf("\n");
    s1=(struct HSstrtype *)malloc(sizeof(struct HSstrtype));      //创建结构体的存储空间
    s1->str=(char *)malloc(MAXNUM);                               //创建字符串的存储空间
    gets(s1->str);
    for(s1->length=0;*s1->str;s1->str++)
        s1->length++;                                             //计算字符串的长度
    for(s1->str=s1->str-s1->length;*s1->str;s1->str++)
        printf("%c",*s1->str);
    printf("%d",s1->length);
    }
```

4.4　文本编辑

文本编辑是串的一个很典型的应用。它被广泛用于各种源程序的输入和修改，也被应用于信函、报刊、公文、书籍的输入、修改和排版。文本编辑的实质就是修改字符数据的形式或格式。在各种文本编辑程序中，它们把用户输入的所有文本都作为一个字符串。尽管各种文本编辑程序的功能可能有强有弱，但是它们的基本的操作都是一致的，一般包括串的输入、查找、修改、删除、输出等。

例如有下列一段源程序：

```
main()
{
  int a,b,c;
  scanf("%d,%d",&a,&b);
  c=a+b;
  printf("%d",c);
}
```

我们把这个源程序看成是一个文本，为了编辑的方便，总是利用换行符把文本划分为若干行，还可以利用换页符将文本组成若干页，这样整个文本就是一个字符串，简称为文本串，其中的页为文本串的子串，行又是页的子串。将它们按顺序方式存入计算机内存中，如表 4-1 所示（图中↙表示回车符）。

表 4-1　　**文本格式示例**

m	a	i	n	(	)	↙			{	i	n	t		a	,	b	,
c	;	↙		s	c	a	n	f	(	〃	%	d	,	%	d	〃	
,	&	a	,	&	b	;	↙			c	=	a	+	b	;	↙	
	p	r	i	n	t	f	(	〃	%	d	〃	,	c	)	;	↙	
	}	↙															

注：起始地址为 800。

在输入程序的同时，文本编辑程序先为文本串建立相应的页表和行表，即建立各子串的存储映像。串值存放在文本工作区，而将页号和该页中的起始行号存放在页表中，行号、串值的存储起始地址和串的长度记录在行表，由于使用了行表和页表，因此新的一页或一行可存放在文本工作区的任何一个自由区中，页表中的页号和行表中的行号是按递增的顺序排列的，如表 4-2 所示。设程序的行号从 110 开始。

表 4-2 行表及其信息排列

行号	起始地址	长度
110	800	7
120	809	12
130	823	21
140	846	7
150	855	16
160	873	2

下面我们就来讨论文本的编辑。

(1)插入一行时,首先在文本末尾的空闲工作区写入该行的串值,然后,在行表中建立该行的信息,插入后,必须保证行表中行号从小到大的顺序。若插入行 145,则行表中从 150 开始的各行信息必须向下平移一行。

(2)删除一行时,则只要在行表中删除该行的行号,后面的行号向前平移。若删除的行是页的起始行,则还要修改相应页的起始行号(改为下一行)。

(3)修改文本时,在文本编辑程序中设立了页指针、行指针和字符指针,分别指示当前操作的页、行和字符。若在当前行内插入或删除若干字符,则要修改行表中当前行的长度。如果该行的长度超出了分配给它的存储空间,则应为该行重新分配存储空间,同时还要修改该行的起始位置。

对页表的维护与行表类似,在此不再叙述,有兴趣的同学可设计其中的算法。

4.5 实训项目四 成绩管理系统

【实训】成绩管理系统

1.实训说明

本实训是关于串的应用,在本实训中主要利用串的链式存储结构,对学生的各项记录动态的存储,并且将结果保存在文件中,可以调用以前的数据。从而加深对串的基本存储方法和基本运算的了解,以及简单的文件操作。

设计要求:可以完成学生数据的输入输出,并进行简单的管理。

要求实现以下的基本功能模块:

(1)输入学生成绩;(2)删除学生成绩;(3)显示所有学生;

(4)保存为文本文件;(5)从文件读取。

完成以上模块后,有兴趣可以考虑以下功能模块的实现:

(1)将文件进行复制;(2)进行排序;(3)将学生成绩追加到文本文件;

(4)进行分类汇总。

2.程序分析

采用链式存储方式,要定义一个结构体:

```
typedef struct z1    /*定义数据结构*/
{
   char no[11];
   char name[15];
   int score[N];
   float sum;
   float average;
   int order;
   struct z1 *next;
}STUDENT;
```

定义以下函数:

- STUDENT　*init():初始化函数。
- STUDENT *create():创建链表,输入学生数据,当学号为@时,停止输入。
- STUDENT *delete(STUDENT *h):删除记录,根据学号进行删除,删除成功返回头指针。
- void print(STUDENT *h):显示所有记录。
- void save(STUDENT *h):以文本文件保存学生成绩,输入文件名要标明路径,如没有该文件则自动创建一个新文件。
- STUDENT *load():读取记录。

3.程序源代码

```
#include "stdio.h"     /*I/O函数*/
#include "stdlib.h"    /*其他说明*/
#include "string.h"    /*字符串函数*/
#include "conio.h"     /*屏幕操作函数*/
#include "mem.h"       /*内存操作函数*/
#include "ctype.h"     /*字符操作函数*/
#include "alloc.h"     /*动态地址分配函数*/
#define N 3            /*定义常数*/
typedef struct z1    /*定义数据结构*/
{
   char no[11];
   char name[15];
   int score[N];
   float sum;
   float average;
   int order;
   struct z1 *next;
}STUDENT;
STUDENT  *init(); /*初始化函数*/
STUDENT *create(); /*创建链表*/
```

```
STUDENT *delete(STUDENT *h);   /*删除记录*/
void print(STUDENT *h);/* 显示所有记录*/
void search(STUDENT *h);/*查找*/
void save(STUDENT *h);/*保存*/
STUDENT *load();/*读入记录*/
int menu_select();/*菜单函数*/
main()
{
   int i;
   STUDENT *head;      /*链表定义头指针*/
   head=init();          /*初始化链表*/
   clrscr();             /*清屏*/
   for(;;)               /*无限循环*/
   {
      switch(menu_select())     /*调用主菜单函数,返回值整数作开关语句的条件*/
      {                        /*值不同,执行的函数不同,break 不能省略*/
        case 0:head=init();break;   /*执行初始化*/
        case 1:head=create();break; /*创建链表*/
        case 2:head=delete(head);break; /*删除记录*/
        case 3:print(head);break;   /*显示全部记录*/
        case 4:search(head);break;    /*查找记录*/
        case 5:save(head);break;     /*保存文件*/
        case 6:head=load();break;    /*读文件*/
        case 7:exit(0);              /*如菜单返回值为 7 程序结束*/
      }
   }
}
menu_select()
{
   char *menu[]={"*********MENU*********",  /*定义菜单字符串数组*/
   " 0. init list",                          /*初始化*/
   " 1. Enter list",                        /*输入记录*/
   " 2. Delete a record from list",          /*从表中删除记录*/
   " 3. print list ",                       /*显示单链表中所有记录*/
   " 4. Search record on name",             /*按照姓名查找记录*/
   " 5. Save the file",                      /*将单链表中记录保存到文件中*/
   " 6. Load the file",                     /*从文件中读入记录*/
   "7. Quit"};                              /*退出*/
   char s[3];                               /*以字符形式保存选择号*/
   int c,i;                                 /*定义整形变量*/
   gotoxy(1,25);                            /*移动光标*/
   printf("press any key enter menu.....\n");   /*压任一键进入主菜单*/
```

```
    getch();                                    /* 输入任一键 */
    clrscr();                                   /* 清屏幕 */
    gotoxy(1,1);                                /* 移动光标 */
    textcolor(YELLOW);                          /* 设置文本显示颜色为黄色 */
    textbackground(BLUE);                       /* 设置背景颜色为蓝色 */
    gotoxy(10,2);                               /* 移动光标 */
    putch(0xc9);                                /* 输出左上角边框 ┌ */
    for(i=1;i<44;i++)
        putch(0xcd);     /* 输出上边框水平线 */
    putch(0xbb);                /* 输出右上角边框  ┐ */
    for(i=3;i<20;i++)
    {
        gotoxy(10,i);putch(0xba);   /* 输出左垂直线 */
        gotoxy(54,i);putch(0xba);
    }   /* 输出右垂直线 */
    gotoxy(10,20);putch(0xc8);  /* 输出左上角边框 └ */
    for(i=1;i<44;i++)
        putch(0xcd);        /* 输出下边框水平线 */
    putch(0xbc);                    /* 输出右下角边框┘ */
    window(11,3,53,19);         /* 制作显示菜单的窗口,大小根据菜单条数设计 */
    clrscr();                 /* 清屏 */
    for(i=0;i<16;i++)           /* 输出主菜单数组 */
    {
        gotoxy(10,i+1);
        cprintf("%s",menu[i]);
    }
    textbackground(BLACK);     /* 设置背景颜色为黑色 */
    window(1,1,80,25);          /* 恢复原窗口大小 */
    gotoxy(10,21);          /* 移动光标 */
    do{
        printf("\n Enter you choice(0～6,14(exit)):");     /* 在菜单窗口外显示提示信息 */
        scanf("%s",s);              /* 输入选择项 */
        c=atoi(s);                /* 将输入的字符串转化为整形数 */
    }while(c<0||c>14);      /* 选择项不在 0～14 之间重输 */
    return c;                   /* 返回选择项,主程序根据该数调用相应的函数 */
}
STUDENT *init()
{
    return NULL;
}

/* 创建链表 */
```

```
STUDENT *create()
{
   int i; int s;
   STUDENT *h=NULL,*info;  /* STUDENT指向结构体的指针*/
   for(;;)
   {
      info=(STUDENT *)malloc(sizeof(STUDENT));  /*申请空间*/
      if(! info)   /*如果指针info为空*/
      {
        printf("\nout of memory");   /*输出内存溢出*/
        return NULL;              /*返回空指针*/
      }
      inputs("enter no:",info->no,11);   /*输入学号并校验*/
      if(info->no[0]=='@') break;     /*如果学号首字符为@则结束输入*/
      inputs("enter name:",info->name,15); /*输入姓名,并进行校验*/
      printf("please input %d score\n",N);  /*提示开始输入成绩*/
      s=0;          /*计算每个学生的总分,初值为0*/
      for(i=0;i<N;i++)     /*N门课程循环N次*/
      {
        do{
            printf("score%d:",i+1);     /*提示输入第几门课程*/
            scanf("%d",&info->score[i]);     /*输入成绩*/
            if(info->score[i]>100||info->score[i]<0) /*确保成绩在0~100之间*/
            printf("bad data,repeat input\n");  /*出错提示信息*/
         }while(info->score[i]>100||info->score[i]<0);
        s=s+info->score[i];   /*累加各门课程成绩*/
      }
      info->sum=s;        /*将总分保存*/
      info->average=(float)s/N;  /*求出平均值*/
      info->order=0;      /*未排序前此值为0*/
      info->next=h;   /*将头结点做为新输入结点的后继结点*/
      h=info;     /*新输入结点为新的头结点*/
   }
   return(h);   /*返回头指针*/
}
/*输入字符串,并进行长度验证*/
inputs(char *prompt, char *s, int count)
{
   char p[255];
   do{
      printf(prompt);  /*显示提示信息*/
      scanf("%s",p);  /*输入字符串*/
```

```
        if(strlen(p)>count)printf("\n too long! \n"); /* 进行长度校验,超过 count 值重输入 */
    }while(strlen(p)>count);
    strcpy(s,p); /* 将输入的字符串拷贝到字符串 s 中 */
}
/* 输出链表中结点信息 */
void print(STUDENT *h)
{
    int i=0;      /* 统计记录条数 */
    STUDENT *p;   /* 移动指针 */
    clrscr();     /* 清屏 */
    p=h;          /* 初值为头指针 */
    printf("\n\n\n********** STUDENT **********\n");
    printf("|rec|no        |      name     | sc1| sc2| sc3|   sum   |  ave  |order|\n");
    printf("|---|------|-----|-----|-----|-------|-------|\n");
    while(p!=NULL)
    {
        i++;
        printf("|%3d|%-10s|%-15s|%4d|%4d|%4d|%4.2f|%4.2f|%3d|\n", i, p->no,p
        ->name,p->score[0],p->score[1],p->score[2],p->sum,p->average,p->order);
        p=p->next;
    }
    printf("**********end**********\n");
}
/* 删除记录 */
STUDENT *delete(STUDENT *h)
{
    STUDENT *p,*q;  /* p 为查找到要删除的结点指针,q 为其前驱指针 */
    char s[11];     /* 存放学号 */
    clrscr();       /* 清屏 */
    printf("please deleted no\n");        /* 显示提示信息 */
    scanf("%s",s);  /* 输入要删除记录的学号 */
    q=p=h;          /* 给 q 和 p 赋初值头指针 */
    while(strcmp(p->no,s)&&p!=NULL)   /* 当记录的学号不是要找的,或指针不为空时 */
    {
        q=p;        /* 将 p 指针值赋给 q 作为 p 的前驱指针 */
        p=p->next;  /* 将 p 指针指向下一条记录 */
    }
    if(p==NULL)     /* 如果 p 为空,说明链表中没有该结点 */
        printf("\nlist no %s student\n",s);
    else        /* p 不为空,显示找到的记录信息 */
    {
        printf("**********have found**********\n");
```

```
        printf("|no    |    name   | sc1| sc2| sc3|   sum   |  ave  |order|\n");
        printf("|---|-----|---|---|-----|---|--|\n");
        printf("| %-10s| %-15s| %4d| %4d| %4d|  %4.2f | %4.2f | %3d |\n", p->no,
         p->name,p->score[0],p->score[1],p->score[2],p->sum,
         p->average,p->order);
        printf("* * * * * * * * * * * * end* * * * * * * * * * * *\n");
        getch();        /* 压任一键后,开始删除 */
        if(p==h)     /* 如果 p==h,说明被删结点是头结点 */
        h=p->next;      /* 修改头指针指向下一条记录 */
        else
        q->next=p->next; /* 不是头指针,将 p 的后继结点作为 q 的后继结点 */
        free(p);            /* 释放 p 所指结点空间 */
        printf("\n have deleted No %s student\n",s);
        printf("Don't forget save\n");/* 提示删除后不要忘记保存文件 */
    }
    return(h);       /* 返回头指针 */
}
STUDENT  *insert(STUDENT *h)  /* 插入记录 */
{
    STUDENT *p,*q,*info; /* p 指向插入位置,q 是其前驱,info 指新插入记录 */
    char s[11];  /* 保存插入点位置的学号 */
    int s1,i;
    printf("please enter location  before the no\n");
    scanf("%s",s);   /* 输入插入点学号 */
    printf("\nplease new record\n");          /* 提示输入记录信息 */
    info=(STUDENT *)malloc(sizeof(STUDENT));   /* 申请空间 */
    if(! info)
    {
        printf("\nout of memory");   /* 如没有申请到,内存溢出 */
        return NULL;               /* 返回空指针 */
    }
    inputs("enter no:",info->no,11); /* 输入学号 */
    inputs("enter name:",info->name,15); /* 输入姓名 */
    printf("please input %d score \n",N);  /* 提示输入分数 */
    s1=0;     /* 保存新记录的总分,初值为 0 */
    for(i=0;i<N;i++)     /* N 门课程循环 N 次输入成绩 */
    {
        do{          /* 对数据进行验证,保证在 0～100 之间 */
printf("score%d:",i+1);
scanf("%d",&info->score[i]);
if(info->score[i]>100||info->score[i]<0)
    printf("bad data,repeat input\n");
```

```
        }while(info->score[i]>100||info->score[i]<0);
        s1=s1+info->score[i];     /*计算总分*/
    }
    info->sum=s1;     /*将总分存入新记录中*/
    info->average=(float)s1/N;  /*计算均分*/
    info->order=0;          /*名次赋值 0*/
    info->next=NULL;      /*设后继指针为空*/
    p=h;                  /*将指针赋值给 p*/
    q=h;                 /*将指针赋值给 q*/
    while(strcmp(p->no,s)&&p!=NULL)     /*查找插入位置*/
    {
        q=p;                    /*保存指针 p,作为下一个 p 的前驱*/
        p=p->next;              /*将指针 p 后移*/
    }
    if(p==NULL)        /*如果 p 指针为空,说明没有指定结点*/
        if(p==h)       /*同时 p 等于 h,说明链表为空*/
        h=info;   /*新记录则为头结点*/
        else
        q->next=info;  /*p 为空,但 p 不等于 h,将新结点插在表尾*/
         else
        if(p==h)      /*p 不为空,则找到了指定结点*/
        {
          info->next=p; /*如果 p 等于 h,则新结点插入在第一个结点之前*/
          h=info;     /*新结点为新的头结点*/
        }
        else
        {
          info->next=p;  /*不是头结点,则是中间某个位置,新结点的后继为 p*/
          q->next=info;  /*新结点作为 q 的后继结点*/
        }
        printf("\n--have inserted %s student--\n",info->name);  printf("--Don't forget
        save--\n");      /*提示存盘*/
    return(h);           /*返回头指针*/
}
/*保存数据到文件*/
void save(STUDENT *h)
{
    FILE *fp;          /*定义指向文件的指针*/
    STUDENT *p;     /*定义移动指针*/
    char outfile[10];  /*保存输出文件名*/
    printf("Enter outfile name,for example c:\\fl\\te.txt:\n"); /*提示文件名格式信息*/
    scanf("%s",outfile);
```

```
    if((fp=fopen(outfile,"wb"))==NULL) /*为输出打开一个二进制文件,如没有则建立*/
    {
        printf("can not open file\n");
        exit(1);
    }
    printf("\nSaving file.....\n");  /*打开文件,提示正在保存*/
    p=h;                          /*移动指针从头指针开始*/
    while(p!=NULL)              /*如p不为空*/
    {
        fwrite(p,sizeof(STUDENT),1,fp);/*写入一条记录*/
        p=p->next;              /*指针后移*/
    }
    fclose(fp);           /*关闭文件*/
    printf("-----save success!! -----\n");  /*显示保存成功*/
}
/*从文件读数据*/
STUDENT *load()
{
    STUDENT *p,*q,*h=NULL;     /*定义记录指针变量*/
    FILE *fp;                 /*定义指向文件的指针*/
    char infile[10];          /*保存文件名*/
    printf("Enter infile name,for example c:\\f1\\te.txt:\n");
       scanf("%s",infile);              /*输入文件名*/
    if((fp=fopen(infile,"rb"))==NULL)  /*打开一个二进制文件,为读方式*/
    {
        printf("can not open file\n");     /*如不能打开,则结束程序*/
        exit(1);
    }
    printf("\n -----Loading file! -----\n");
    p=(STUDENT *)malloc(sizeof(STUDENT));   /*申请空间*/
    if(!p)
    {
        printf("out of memory! \n");     /*如没有申请到,则内存溢出*/
        return h;           /*返回空头指针*/
    }
    h=p;              /*申请到空间,将其作为头指针*/
    while(!feof(fp))   /*循环读数据直到文件尾结束*/
    {
        if(1!=fread(p,sizeof(STUDENT),1,fp))
break;  /*如果没读到数据,跳出循环*/
        p->next=(STUDENT *)malloc(sizeof(STUDENT));  /*为下一个结点申请空间*/
        if(!p->next)
```

```
        {
printf("out of memory! \n");  /* 如没有申请到,则内存溢出 */
return h;
        }
        q=p;   /* 保存当前结点的指针,作为下一结点的前驱 */
        p=p->next;      /* 指针后移,新读入数据链到当前表尾 */
    }
    q->next=NULL;   /* 最后一个结点的后继指针为空 */
    fclose(fp);          /* 关闭文件 */
    printf("---You have success read data from file!!! ---\n");
    return h;           /* 返回头指针 */
}
```

最后程序运行结果如下所示：

```
* * * * * * * * * MENU * * * * * * * * * * * * * *
0. Init list
1. Enter list
2. Delete a record from list
3. Print list
4. Search record on name
5. Save the file
6. Load the file
7. Quit
Enter you choice(0~7):
```

本章小结

本章主要介绍了如下一些基本概念：

串：串(或字符串)(String)是由零个或多个字符组成的有限序列。

主串和子串：一个串的任意个连续的字符组成的子序列称为该串的子串，包含该子串的串称为主串。

串的静态存储结构：类似于线性表的顺序存储结构，用一组地址连续的存储单元存储串值的字符序列的存储方式称为串的顺序存储结构。

堆存储结构：用一组空间足够大的、地址连续的存储单元存放串值字符序列，但其存储空间在程序执行过程中能动态分配的存储方式称为堆存储结构。

串的链式存储结构：类似于线性表的链式存储结构，采用链表方式存储串值字符序列的存储方式称为串的顺序存储结构。

串名的存储映像：串名的存储映像就是建立串名和串值之间的对应关系的一个符号表。

除上述基本概念以外，学生还应该了解串的基本运算、字符串拷贝(赋值、字符串的联接、求字符串的长度、子串的查询、字符串的比较)、串的静态存储结构的表示、串的链式存储结构的表

示、串的堆存储结构的表示,能在各种存储结构方式中求字符串的长度、能在各种存储结构方式中利用C语言提供的串函数进行操作。

习题四

1.简述空串与空格串、串变量与串常量、主串与子串、串名与串值每对术语的区别?

2.两个字符串相等的充分条件是什么?

3.串有哪几种存储结构?

4.已知两个串:s1="fg cdb cabcadr", s2="abc", 试求两个串的长度,判断串s2是否是串s1的子串,并指出串s2在串s1中的位置。

5.已知:s1="I'm a student",s2="student",s3="teacher",试求下列各运算的结果:

```
strstr(s1,s2);
strlen(s1);
strcat(s2,s3);
delstr(s1,4,10);
insstr(str1,7,s3)。
```

6.设s1="AB",s2="ABCD",s3="EFGHIJK",试画出堆存储结构下的存储映像图。

7.试写出将字符串s2中的全部字符拷贝到字符串s1中的算法,不允许利用库函数strcpy()。