

第 4 章

串

本章学习导读

本章主要介绍串的定义及基本运算,重点介绍串的存储结构,基本运算与串的 C# 实现方法。读者学习本章后应能掌握串的定义,串的基本运算,运用串来实现文本输入和输出。

4.1 串的基本概念

4.1.1 串的定义

字符串在应用程序中的使用非常频繁。字符串简称串,是一种特殊的线性表,其特殊性在于串中的数据元素是一个个的字符。在事务处理程序中,顾客的信息如姓名、地址等及货物的名称、产地和规格等,都被作为字符串来处理。另外,字符串还具有自身的一些特性。因此,把字符串作为一种数据结构来研究。

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:

$$S = "c_0 c_1 \cdots c_{n-1}" \quad (n \geq 0)$$

其中,S 是串名,双引号作为串的定界符(不是串的一部分),用双引号引起来的字符序列是串值。 $c_i (0 \leq i \leq n)$ 可以是字母、数字、空格或其他字符, n 为串的长度,当 $n=0$ 时,称为空串(Empty String)。在计算串长时,空格也应记入串的长度中,如 $S = "I'm a student"$ 的长度为 13。

4.1.2 主串和子串

一个串中任意个连续的字符组成的子序列称为该串的子串(Substring)。包含子串的串相应地称为主串。子串的的第一个字符在主串中的位置叫子串的位置。如串 $s = "I'm a student"$, 它的长度是 13,串 $s_1 = "student"$ 的长度是 7, s_1 是 s 的子串, s_1 的位置是 7。

如果两个串的长度相等并且对应位置的字符都相等,则称这两个串相等。而在 C# 中,比较两个串是否相等还要看串的语言文化等信息。

4.2 串的存储结构

4.2.1 串值的存储

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以赋值给一个串变量,操作时通过串变量名访问串值。

由于串中的字符都是连续存储的,而在 C# 中串具有恒定不变的特性,即字符串一经创建,就不能将其变长、变短或者改变其中任何的字符。所以,这里不讨论串的链式存储,也不用接口来表示串的操作。同样,把串看作是一个类,类名为 StringDS。取名为 StringDS 是为了和 C# 自身的字符串类 String 相区别。类 StringDS 只有一个字段,即存放串中字符序列的数组 data。由于串的运算有很多,在类 StringDS 中只包含部分基本的运算。串类 StringDS 的 C# 实现如下所示:

```
public class StringDS
{
    private char[] data;           //字符数组
    public char this[int index]   //通过数组下标来访问
    {
        get
        {
            return data[index];
        }
    }
    public StringDS(char[] arr)   //构造器,构造一个方法
    {
        data=new char[arr.Length];
        for(int i=0; i < arr.Length; ++i)
        {
            data[i]=arr[i];      //实现字符数组的赋值
        }
    }
    public StringDS(StringDS s)
    {
        for(int i=0; i < arr.Length; ++i)
        {
            data[i]=s[i];
        }
    }
}
```

```
//构造器
public StringDS(StringDS s)
{
    for(int i=0; i < arr.Length; ++i)
    {
        data[i]=s[i];
    }
}
//构造器
public StringDS(int len)
{
    char[] arr=new char[len];
    data=arr;
}
//求串长
public int GetLength()
{
}
//串比较
public int Compare(StringDS s)
{
}
//求子串
public StringDS SubString(int index, int len)
{
}
//串连接
public StringDS Concat(StringDS s)
{
}
//串插入
public StringDS Insert(int index, StringDS s)
{
}
//串删除
public StringDS Delete(int index, int len)
{
}
//串定位
public int Index(StringDS s)
{
}
```

说明:以上 C# 程序只是部分代码,主要介绍在 C# 如何在一个类中通过定义一个字符数组来存储字符串,并通过下标的方法来对该字符数组进行赋值操作。具体完整的操作字符串的 C# 程序将在后面的字符串基本运算及实现仔细说明。

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 4-1 所示。

地址	800	803	804								812							
串值	d	a	t	a	s	t	r	u	c	t	u	r	e					

图 4-1 串 s1 和 s2 的存储状态

若符号表中每行包含有串名、串值的始地址、尾地址,则如图 4-2(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 4-2(b)所示。

串名	始地址	尾地址	串名	始地址	串长
S1	800	803	S1	800	4
S2	804	812	S2	804	9

(a)

(b)

图 4-2 符号表示例

4.3 串的基本运算及其实现

串的基本运算有赋值、连接、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,将使用 C# 程序来介绍这些串运算(串操作)。

4.3.1 串的基本运算

(1)GetLength(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。

(2)Compare(StringDS s) 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。

(3)SubStr(int index, int len) 求子串:从主串的 index 位置起找长度为 len 的子串,若找到,返回该子串;否则,返回空值 NULL。

(4)Str_Concat(StringDS s2) 串的连接:把字符串 str2 接到 str1 后面,str1 最后的结尾符

'\0'被取消。返回 str1。

(5)Str_Inst(int index, StringDS s2) 字符串的插入:在字符串 str1 第 index 个位置之前开始,插入字符串 str2。返回 str1。

(6>Delete(int index, int len) 字符串的删除:在字符串 str 中,删除从第 index 个字符开始的 len 个长度的子串。

(7)Index(StringDS s) 字符串的定位:查找子串 s 在主串中首次出现的位置。如果找到,返回子串 s 在主串中首次出现的位置,否则,返回-1。

4.3.2 串的基本运算实现

1. 求串的长度

求串的长度就是求串中字符的个数,可以通过求数组 str 的长度来求串的长度。求串的长度实现如下:

```
public int GetLength(str)
{
    return str.Length;    //通过数组的 Length 属性来返回串的长度
}
```

2. 串的比较

如果两个串的长度相等并且对应位置的字符相同,则串相等,返回 0;如果串 s1 对字符大于串 s2 应位置的字符返回 1,若小于,则返回-1。

说明:需要注意,使用 string.Equals 来比较两个串是否相等的,如,s1.Equals(s2)返回结果为布尔值真或假。而使用 string.Compare 来比较字符串时,返回值有三种情况:<0 小于,=0 等于,>0 大于。

具体实现程序如下:

```
public int Compare(StringDS s)
{
    int len=((this.GetLength()<=s.GetLength())? this.GetLength():s.GetLength());
    int i=0;
    for (i=0; i<len; ++i)
    {
        if (this[i] != s[i])
        {
            break;
        }
    }
    if (i <= len)
    {
        if (this[i] < s[i])
        {
            return -1;
        }
    }
}
```

```
    }
    else if (this[i] > s[i])
    {
        return 1;
    }
}

else if (this.GetLength() == s.GetLength())
{
    return 0;
}

else if (this.GetLength() < s.GetLength())
{
    return -1;
}

return 1;
}
```

3. 求子串

从主串的 index 位置起找长度为 len 的子串,若找到,返回该子串,否则,返回一个空串。具体算法实现如下:

```
public StringDS SubStr(int index, int len) //len为子串的长度,index开始取的位置
{
    if((index<0)||((index>this.GetLength()-1)||((len<0)||((len>this.GetLength()-index)))
    {
        Console.WriteLine("Position or Length is error!");
        return null;
    }
    StringDS substr=new StringDS(len); //创建一个字符串 substr 用来存储取出的字符
    for(int i=0; i<len; ++i)
    {
        substr[i]=this[i+index-1];
    }
    return substr;
}
```

4. 字符串的连接

串连接的算法实现如下:

```
public StringDS Str_Concat(StringDS s2)
{
    StringDS s1=new StringDS(this.GetLength()+ s2.GetLength()); //连接后串的长度
    for(int i=0; i<this.GetLength(); ++i)
    {
        s1.data[i]=this[i]; //s1 原来的部分
```

```
    }  
    for(int j= 0; j<s2.GetLength();++j)  
    {  
        s1.data[this.GetLength()+j]=s2[j];    // 将 s2 连接到 s1 之后  
    }  
    return s1;  
}
```

5. 字符串的插入

字符串的插入是在一个串 s1 的某个位置如 index 处插入一个串 s2。如果位置符合条件,则该操作返回一个新串,新串的长度是串 s1 的长度与串 s2 的长度之和,新串的第 1 部分是该串的开始字符到第 index 之间的字符,第 2 部分是串 s2,第 3 部分是该串从 index 位置字符到该串的结束位置处的字符。如果位置不符合条件,则返回一个空串。

字符串插入的算法如下:

```
public StringDSStr_Inst(int index, StringDS s2)  
{  
    int len=s2.GetLength();  
    int len2=len+this.GetLength(); //插入 s2 后新串的长度  
    StringDS s1=new StringDS(len2);  
    if(index<0||index>this.GetLength()-1)  
    {  
        Console.WriteLine("Position is error!");  
        return null;  
    }  
    for(int i=0; i<index; ++i)  
    {  
        s1[i]=this[i];    //插入点 index 位置之前的部分  
    }  
    for(int i=index; i<index+len; ++i)  
    {  
        s1[i]=s2[i-index];    //将 s2 字符串插入到 s1 中  
    }  
    for(int i=index+len; i<len2; ++i)  
    {  
        s1[i]=this[i-len];    //原字符串中 index 位置之后的部分  
    }  
    return s1;  
}
```

6. 字符串的删除

字符串删除是把一个串中从第 index 位置起连续的 len 长度个字符的子串从主串中删除掉。如果位置和长度符合条件,则该操作返回一个新串,新串的长度是原串的长度减去 len,新串的前部分是原串的开始到第 index 个位置之间的字符,后部分是原串从第 index+len 位置到

原串结束的字符。如果位置和长度不符合条件,则返回一个空串。

串删除的算法实现如下:

```
public StringDS Delete(int index, int len)
{
    if ((index<0)|| (index>this.GetLength()-1) || (len<0)|| (len>this.GetLength()-index))
    {
        Console.WriteLine("Position or Length is error!");
        return null;
    }
    StringDS s=new StringDS(this.GetLength()-len);
    for(int i =0; i<index; ++i)
    {
        s[i]=this[i];
    }
    for (int i =index+len; i<this.GetLength(); ++i)
    {
        s[i]=this[i];
    }
    return s;
}
```

7. 字符串的定位

串定位的算法实现如下:

```
public int Index(StringDS s)
{
    if(this.GetLength() < s.GetLength())
    {
        Console.WriteLine("There is not string s!");
        return -1;
    }
    int i=0;
    int len =this.GetLength()-s.GetLength();
    while(i<len)
    {
        if (Compare(s)==0)
        {
            break;
        }
    }
    if (i <= len)
    {
        return i;
    }
    return -1;
}
```

在C#中,一个String表示一个恒定不变的字符序列集合。String类型是封闭类型,所以,

它不能被其他类继承,而它直接继承自 object。因此,String 是引用类型,不是值类型,在托管堆上而不是在线程的堆栈上分配空间。String 类型还继承了 IComparable、ICloneable、IConvertible、IComparable<string>、IEnumerable<char>、IEnumerable 和 IEquatable<string>等接口。String 的恒定性指的是一个串一旦被创建,就不能将其变长、变短或者改变其中任何的字符。所以,当我们对一个串进行操作时,不能改变字符串,如在本教材定义的 StringDS 类中,串连接、串插入和串删除等操作的结果都是生成了新串而没有改变原串。C# 也提供了 StringBuilder 类型来支持高效地动态创建字符串。

在 C# 中,创建串不能用 new 操作符,而是使用一种称为字符串驻留的机制。这是因为 C# 语言将 String 看作是基元类型。基元类型是被编译器直接支持的类型,可以在源代码中用文本常量(Literal)来直接表达字符串。当 C# 编译器对源代码进行编译时,将文本常量字符串存放在托管模块的元数据中。而当 CLR 初始化时,CLR 创建一个空的散列表,其中的键是字符串,值为指向托管堆中字符串对象的引用。散列表就是哈希表,关于散列表的详细介绍见 8.4 小节。当 JIT 编译器编译方法时,它会在散列表中查找每一个文本字符串常量。如果找不到,就会在托管堆中构造一个新的 String 对象(指向字符串),然后将该字符串和指向该字符串对象的引用添加到散列表中;如果找到了,不会执行任何操作。

C# 提供的 String 类型中的方法很多,比如构造器有 8 个,比较两个字符串的方法有 12 个,连接字符串的方法有 9 个。以下列出了该类型中常用的方法,并对每个方法给出了注释。关于 String 更为详细的介绍参考.NET 的有关书籍。

```
public sealed class String: IComparable, ICloneable, IConvertible,
    IComparable<string>, IEnumerable<char>, IEnumerable, IEquatable<string>
{
    //将串初始化为由字符数组指示的值。
    public String(char[] value);
    //确定两个指定的 String 对象是否具有不同的值。
    public static bool operator !=(string a, string b);
    //确定两个指定的 String 对象是否具有同一值。
    public static bool operator ==(string a, string b);
    //返回对此串的引用。
    public object Clone();
    //返回与指定的串的排序情况。
    public int CompareTo(string strB);
    //连接两个串。
    public static string Concat(string str0, string str1);
    //创建一个与指定的串具有相同值的串。
    public static string Copy(string str);
    public bool EndsWith(string value, StringComparison comparisonType);
    //返回指定的串第一个匹配项的索引。
    public int IndexOf(string value);
    //在串的指定索引位置插入一个串。
    public string Insert(int startIndex, string value);
    //将串的所有匹配项替换为其他指定的串
    public string Replace(string oldValue, string newValue);
```

```
public string Substring(int startIndex, int length); //从指定的字符位置检索子字符串
public string ToLower(); //将串中的字符转换为小写形式。
public string ToUpper(); //将串中的字符转换为大写形式。
.....
}
```

4.4 实训项目四——学生成绩管理系统

【实训】学生成绩管理系统。

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,要定义一个学生类:

```
public class Student /* 定义数据结构 */
{
    public int no;
    public double math;
    public double english;
    public double csharp;
    public double network;
    public double avg, all;
    public string name;
}
```

定义以下方法函数:

(1)public void ShowStuInfo();

显示学生信息。

(2)public void Add();

添加学生信息。

(3)public void Delete();

删除学生信息。

(4)public void Search_no();

查询学生信息(按学号查询)。

(5)public void Search_name();

查询学生信息(按姓名查询)。

(6)public void Search_score();

查询学生成绩。

(7)public void Chech();

选择菜单。

系统以文本文件保存学生成绩,输入菜单项前的数组符号进入对应的操作项,实现对应的功能。

3. 程序源代码

```
using System; //各种C#包(命名空间)的引入
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace 学生信息管理
{
publicclass Student
{ public int no;
public double math, english;
public double csharp, network;
public double avg, all;
public string name;
public void ShowStuInfo() //显示学生信息
{
Console.WriteLine("\n 学生信息:");
Console.WriteLine("学号:{0}", no);
Console.WriteLine("\t 姓名 :{1}", name);
Console.WriteLine("\t 数学 :{2}", math);
Console.WriteLine("\t 英语 :{3}", english);
Console.WriteLine("\t C# :{4}", csharp);
Console.WriteLine("\t 计算机网络 :{5}", network);

Console.WriteLine("\t 平均成绩 :{6}", avg);
Console.WriteLine("\t 总成绩 : {7}", all);
Console.WriteLine();
}
}
}
class Stu_Manage //定义学生信息管理类
{
int x=0;
```

```
Student[] stud=new Student[10];    //声明一个结构体类数组,数组名 stud
public void Add()                    // Add()方法添加学生信息
{
    Console.WriteLine("请输入要添加的学生信息:");
    Console.WriteLine("\n 请输入学号:");
    stud[x].no=int.Parse(Console.ReadLine());
    Console.WriteLine("\n 请输入姓名:");
    stud[x].name=Console.ReadLine();
    Console.WriteLine("\n 数学成绩:");
    stud[x].math=double.Parse(Console.ReadLine());
    Console.WriteLine("\n 英语成绩:");
    stud[x].english=double.Parse(Console.ReadLine());
    Console.WriteLine("\n C#成绩:");
    stud[x].csharp=double.Parse(Console.ReadLine());
    Console.WriteLine("\n 计算机网络成绩:");
    stud[x].network=double.Parse(Console.ReadLine());
    double[] inform=new double[] { stud[x].math, stud[x].english, stud[x].csharp, stud
[x].network }; //定义一个数组将输入的成绩存储
    for (int i=0; i < inform.Length; i++)
    {
        stud[x].all += inform[i];
        stud[x].avg=stud[x].all / 4;
        stud[x].ShowStuInfo();
        x=x+1;
    }
}
public void Delete()                //删除学生信息方法
{
    int n=-1;
    int no=int.Parse(Console.ReadLine());
    for (int i=0; i < x; i++)
    {
        if (no==stud[i].no)
        {
            n=i;
            for (int c=n+1; c < x; c++)
            {
                stud[c - 1]=stud[c]; //交换数组索引值,对指定元素进行删除
            }
            x=x - 1;
            break;
        }
    }
}
```

```
}

    public void Search_no()    //按学号查询学生信息
    {
        int n=-1;
        int no=int.Parse(Console.ReadLine());
        for (int i=0; i < x; i++)
        {
            if (no==stud[i].no)
            {
                n=i;
                stud[i].ShowStuInfo();
                break;
            }
        }
        if (n== -1)
        {
            Console.WriteLine("输入学号有误,请重新输入!");
        }
    }

    public void Search_name()    //按姓名查询学生信息
    {
        int n=-1;
        string name=Console.ReadLine();
        for (int i=0; i < x; i++)
        {
            if(name==stud[i].name)
            {
                n=i;
                stud[i].ShowStuInfo();
                break;
            }
        }
        if (n== -1)
        {
            Console.WriteLine("输入姓名有误,请重新输入!");
        }
    }

    public void Search_score()    //查询学生成绩
    {
        for (int i=x - 1; i >= 0; i--)    //使用冒泡法对学生成绩排序
            for (int j=0; j <= i; j++)
```

```
{
    if (stud[j].all < stud[j+1].all)
    {
        stud[x]=stud[j];
        stud[j]=stud[j+1];
        stud[j+1]=stud[x];
    }
}
int[] mc=new int[x];
for (int i=0; i < x; i++) //使用循环输出排序后的学生成绩
{
    mc[i]=i+1;
    Console.Write(mc[i]+"\\t");
    Console.Write(stud[i].no+"\\t");
    Console.Write(stud[i].name+"\\t");
    Console.Write(stud[i].math+"\\t");
    Console.Write(stud[i].english+"\\t");
    Console.Write(stud[i].csharp+"\\t");
    Console.Write(stud[i].network+"\\t");
    Console.Write(stud[i].avg+"\\t");
    Console.Write(stud[i].all+"\\t");
    Console.WriteLine();
}
}

public void Chech() //选择菜单
{
    do
    {
        Console.WriteLine("请选择:\\n 1、添加学生信息 \\n 2、删除学生信息 \\n 3、查询(按学
号) \\n 4、查询(按姓名) \\n 5、查询成绩单 \\n 6、退出系统");
        int number= int.Parse(Console.ReadLine());
        if (number > 6 || number < 1)
        {
            Console.WriteLine("输入有误请重新输入!");
        }
        switch (number)
        {
            case 1:
                Add();
                break;
            case 2:
                Console.WriteLine("请输入要删除学生的学号:");
```

```
        Delete();
        break;
    case 3:
        Console.WriteLine("请输入学号:");
        Search_no();
        break;
    case 4:
        Console.WriteLine("请输入姓名:");
        Search_name();
        break;
    case 5:
        Console.WriteLine("成绩单:");
        Console.WriteLine("名次 \t 学号 \t 姓名 \t 数学 \t 英语
            \t C# \t 计算机网络 \t 平均 \t 总成绩 ");
        Search_score();
        break;
    case 6:
        Environment.Exit(0);
        break;
    }
    Console.WriteLine("\n 是否要继续!");
} while (true);
}
}
class Program
{
    static void Main(string[] args)
    {
        Stu_Manage myuser = new Stu_Manage();
        myuser.Chech();
    }
}
```

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、字符串的连接、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: $s_1 = \text{"fg cdb cabcad"}$, $s_2 = \text{"abc"}$,试求两个串的长度,判断串 s_2 是否是串 s_1 的子串,并指出串 s_2 在串 s_1 中的位置。
- (5)已知: $s_1 = \text{"I'm a student"}$, $s_2 = \text{"student"}$, $s_3 = \text{"teacher"}$,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 $s_1 = \text{"AB"}$, $s_2 = \text{"ABCD"}$, $s_3 = \text{"EFGHIJK"}$,试画出堆存储结构下的存储映像图。
- (7)试写出将字符串 s_2 中的全部字符拷贝到字符串 s_1 中的算法,不允许利用库函数 `strcpy()`。
- (8)设 s_1 和 s_2 是用结点大小为 1 的单链表表示的串,试写出找出 s_2 中第一个不在 s_1 中出现的字符的算法。
- (9)设字符串采用块链存储结构,块链中每个结点存放 m ($m=4$) 个字符,试写出实现字符串删除的算法。