第6章　字符串与文件操作

计算机并不仅仅能够处理数学问题，还可以用来处理文字，比如写文章、处理代码、记录信息等，如果需要将各种语句记录在计算机中，就要用到字符串或者字符数组。

在本书的开始已经尝试输出过“I love Luogu”的字符串，也介绍过单个字符和数字对应的ASCII编码。本章将介绍字符串的存储和处理方法，同时也初步接触STL，这使得我们可以“站在前人的肩膀上”完成程序，简化编程的难度。

本章的最后一部分介绍文件输入输出，包括NOI系列比赛在内的很多比赛的输入输出要求。图6-1所示为本章思维导图。

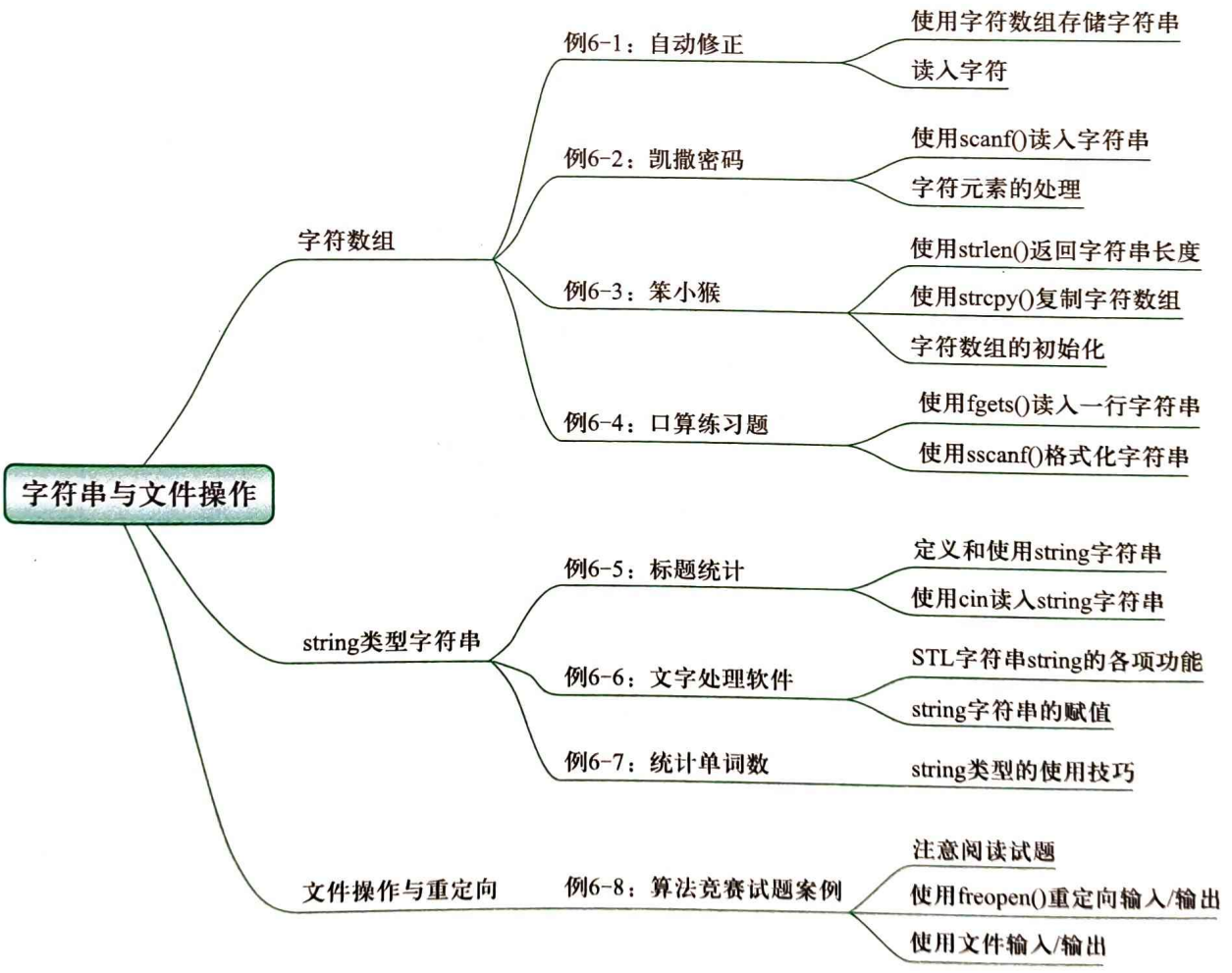

图6-1　本章思维导图

6.1 字符数组

字符数组实质上和整数数组没什么区别，只是数组中的每一个元素都是一个字符（实际上，都是存成对应 ASCII 的数字）。将这些字符存储下来，便组成了一串字符，可以进行进一步操作。为了方便读者，再次放上 ASCII 表（见表 6-1）。当然，读者并不需要把这个表背下来，只需要记住几个重要的对应数字就可以了（比如 '0'、'A'、'a'）。

表 6-1　ASCII 表

数字	对应字符	数字	对应字符	数字	对应字符	数字	对应字符	数字	对应字符
32	[空格]	51	3	70	F	89	Y	108	l
33	!	52	4	71	G	90	Z	109	m
34	"	53	5	72	H	91	[	110	n
35	#	54	6	73	I	92	\	111	o
36	$	55	7	74	J	93	]	112	p
37	%	56	8	75	K	94	^	113	q
38	&	57	9	76	L	95	_	114	r
39	'	58	:	77	M	96	`	115	s
40	(	59	;	78	N	97	a	116	t
41	)	60	<	79	O	98	b	117	u
42	*	61	=	80	P	99	c	118	v
43	+	62	>	81	Q	100	d	119	w
44	,	63	?	82	R	101	e	120	x
45	-	64	@	83	S	102	f	121	y
46	.	65	A	84	T	103	g	122	z
47	/	66	B	85	U	104	h	123	{
48	0	67	C	86	V	105	i	124	|
49	1	68	D	87	W	106	j	125	}
50	2	69	E	88	X	107	k	126	~

例 6-1　自动修正（洛谷 P5733）。大家都知道，一些办公软件有自动将小写字母转换为大写的功能。输入一个长度不超过 100 且不包括空格的字符串。要求将该字符串中的所有小写字母转换成大写字母并输出。例如输入“Luogu4!”，输出“LUOGU4!”。

分析：既然单个字符使用 char 类型存储，回想一下上一章刚刚介绍过的数组，存储一排字符是不是可以使用数组呢？可以定义一个数组，其中每一个元素都是字符类型，这样的**字符数组**就可以存储**字符串**。

有很多办法可以将字符串读入并存进字符串数组中。用 scanf("%s",s);语句读入一个字符串，其中 %s 表示读入的这个数据的类型是字符串，s 是定义的字符数组名（注意，这里的 s 前面没有表示取地址的 &）。类似的可以使用 cin >> s 来读入这个字符串。需要注意的是，这两种读

入方式只能读入到空格或者换行符为止，如果输入包括空格或者换行但是又想读进同一个字符数组中，就要想想别的办法啦。代码实现如下：

```
#include <iostream>
#include <cstdio>
using namespace std;
int main() {
    char s[110];
    scanf("%s", s); // 读入这个字符串，还可以使用 cin>>s; 语句
    for (int i = 0 ; s[i] != '\0' ; i++)
        if ('a' <= s[i] && s[i] <= 'z') /* 如果这个字符在'a'到'z'中间，说明是小写字母 */
            s[i] -= 'a' - 'A'; // 把这个字母转换成对应的大写字母，减去偏移量
    printf("%s\n", s); // 输出这个字符串，还可以使用 cout<<s<<endl; 语句
    return 0;
}
```

观察 ASCII 表，小写字母的顺序和大写字母的顺序分别是按照字母表顺序给出的，'a'-'A' 就是小写字母和对应大写字母的 ASCII 的差距，注意小写字母的 ASCII 编码比较大。如果判断得到这是一个小写字母，就将其减去这个差距，这样得到的 ASCII 值对应的就是大写字母了。

这个字符串实际上在字符数组中的储存方式如图 6-2 所示。

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]
76'L'	117'u'	111'o'	103'g'	117'u'	52'4'	33'!'	0'\0'

图 6-2　字符数组的存储方式

数组 s 中的每一个元素，都储存了一个不超过 127 的整数，这些整数代表了对应的 ASCII 编码字符。最开始的那个字符存储在 s [0] 中。这个字符串虽然只有 7 个字符，但这个字符数组却占了 8 位（s [0] 到 s [7]）。字符串结束后，还需要有一个特殊的“结束标记字符”——'\0'，其对应的数字就是 0。这个结束标记用于提示一个字符串的结束位置，输出时碰到这个标记就停止输出了。顺带一提，类似这样的“特殊字符”还有好几个，除了这里介绍过的 '\0' 还有表示换行的 '\n'，如果要表示一个单引号就写成 '\''。请读者思考一下，如何表示一个反斜杠（\）呢？

当然，也可以不用读入整个字符串，而是每次读入一个字符，判断是否需要处理，再将这个字符直接输出。这里可以使用 getchar( ) 函数获取输入数据中的一个字符（读入后就准备读取接下来的字符了），而相应的，putchar( ) 函数则是输出一个字符。

```
#include <iostream>
#include <cstdio>
using namespace std;
int main() {
    char s;
    while (1) {
```

```
        s = getchar(); // 每次调用 getchar() 函数，读入一个字符
        if (s == EOF) break;
        if ('a' <= s && s <= 'z') // 如果这个字符是小写字母
            s += 'A' - 'a'; // 把它转换成大写字母，这样写和上面是一样的
        putchar(s); // 调用 putchar() 函数，输出一个字符
    }
    return 0;
}
```

运行程序，结果发现无论输入什么，程序都没有反应，这是因为程序不认为输入已经结束了，继续在等待输入。遇到这种情况，输入完字符串后，按一下 Ctrl+Z 组合键，再按一次回车，就可以完成读入了。程序中读入一个字符都会判断是否读完了整个文件，如果文件被读完了，那么 getchar( ) 函数会返回 EOF（一个特殊的常量），即 End of File，这标志着读入已经结束了。在控制台中可以使用 Ctrl+Z 组合键（Windows 下）或者 Ctrl+D 组合键（Linux 下）来输入 EOF 标记提示程序输入已经完毕。

> 至于一些教材使用的 gets( ) 函数将字符串读入字符数组，由于存在字符数组越界的风险，已经不再建议使用，新的 C++11 标准更是删除了这个函数。而输出一个字符串还可以使用 puts( ) 方法，同时会自动输出换行，这倒是还能使用。

例 6-2　凯撒密码（洛谷 P1914）。凯撒密码是由原文字符串（由不超过 50 个小写字母组成）中每个字母向后移动 n 位形成的。z 的下一个字母是 a，如此循环。给出 n 和移动前的原文字符串，请求出密码。

分析：读入这个字符串，然后将每个字符处理后输出。既然往后面移动 n 位，直接在这个字符上加上 n 输出不就可以了吗？并不行，因为加上 n 之后，对应的 ASCII 编码就有可能超出 'a' 到 'z' 的区间，不知道会输出什么了。题目要求 'z' 的下一个是 'a'，如此循环，因此需要思考别的方法。

使用 s[i]-'a' 计算这个小写字母是字母表中的第几个（'a' 是第 0 个，'b' 是第 1 个，确切地说是和 'a' 的偏移量）。然后加上 n，得到目标字母的位置，比如说 'b' 这个字母移动 4 位，就是第 1 个字母向右移动 4 位，是第五个字母，即 'f'。为了要求这个位置始终在 0 到 25 之间，需要对 26 取模。最后还要加回 'a'，变成 ASCII 中对应的字母。代码实现如下：

```
#include <iostream>
#include <cstdio>
using namespace std;
int main() {
    int n;
    char s[60];
    scanf("%d %s", &n, s);    // 读入字符串
    for (int i = 0 ; s[i] ; i++)
        putchar((s[i] - 'a' + n) % 26 + 'a'); // 计算偏移量并还原
    return 0;
}
```

例 6-3　笨小猴(洛谷 P1125,NOIP2008 提高组)。给出一个单词(由不超过 100 个小写字母组成),假设 maxn 是单词中出现次数最多的字母的出现次数,minn 是单词中出现次数最少的字母的出现次数,如果 maxn-minn 是一个质数,那么笨小猴就认为这是个 Lucky Word,输出 Lucky Word,然后在第二行输出 maxn-minn 的值;否则输出 No Answer,第二行输出 0。

分析:读入每个字符串之后,用一个数组记录从 'a' 到 'z' 中每个字母出现的数量(注意这里 a[i]-'a' 可以将字母 'a' 到 'z' 转换为 0 到 25 的数字),然后再使用“打擂台”的思路来寻找出现次数最多字母的次数和最少次数(最少次数不能为 0),最后判断这个差是否为质数即可。需要注意的是,0 和 1 都不是质数。代码如下:

```
#include <cstdio>
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char a[110];
    int ans[26] = {0}; /* ans[0] 到 ans[25] 分别代表 'a' 到 'z' 出现的次数，注意要初始化 */
    int l, mmax, mmin, delta; /* 字符长度，出现次数最多的字母出现次数和出现次数最少的字母出现次数，以及差值 */
    scanf ("%s", a);
    l = strlen (a);
    for (int i = 0; i < l; i++)
        ans[a[i] - 'a']++; // 统计增加某个字母的数量
    mmax = 0; mmin = 10000; // 最大最小值初始化
    for (int i = 0; i < 26; i++) { // 寻找每个字母的最大值和最小值
        if (ans[i] > mmax) mmax = ans[i]; // 如果超过最大值
        if (ans[i] != 0 && ans[i] < mmin) mmin = ans[i]; /* 如果小于最小值，但是不能为 0*/
    }
    delta = mmax - mmin
    if (delta == 0 || delta == 1) { // 质数特判
        printf ("No Answer\n0\n");
        return 0;
    }
    for (int h = 2; h * h <= delta; h++) // 枚举质数
        if (delta % h == 0) {
            printf ("No Answer\n0\n"); // 直接输出答案并退出程序
            return 0;
        }
    printf ("Lucky Word\n%d\n", mmax - mmin);
    return 0;
}
```

这里遇到了新的函数和头文件。首先知道输入的字符串的长度，除了对字符串从前到后枚举 '\0' 的位置以外，还可以使用 strlen 函数来查询字符串的长度。该函数包含在 cstring 头文件里的，这个头文件里面还有其他的一些函数可用于处理字符数组，比如，strcpy 可用来复制字符数组、strcmp 可用于比较两个字符数组（按照字典序）等，这里不再详细叙述。

字符数组如何赋值一个字符串常量呢？不能直接赋值，因为字符数组的数组名也只是一个数组名。可以使用 strcpy( ) 函数来复制一遍字符串常量到字符数组中，例如 char a [100]; strcpy(a,"hello");。类似的，将字符数组 b 的字符串中的内容赋值到字符数组 a，也是不能直接赋值的，必须用 strcpy(a,b)。但是在初始化的时候例外：在初始化的时候可以像初始化数组那样赋值一个字符串常量，例如 char a [100]="Luogu!"。

例 6-4 口算练习题（洛谷 P1957）。王老师收集了 i（$i \leqslant 50$）道学生经常做错的口算题，并且想整理编写成一份练习。王老师希望尽量减少输入的工作量，比如 5+8 的算式最好只输入 5 和 8，输出的结果要尽量详细以方便后期排版使用。对于上述输入进行处理后，输出 5+8=13 以及该算式的总长度 6。

输入数据第 1 行是 i，接着的 i 行是需要输入的算式，每行可能有 3 个数据或两个数据。

1）若该行是 3 个数据，则第一个数据表示运算类型，a 表示加法运算，b 表示减法运算，c 表示乘法运算，接着的两个数据表示参加运算的运算数。

2）若该行是两个数据，则表示本题的运算类型与上一题的运算类型相同，而这两个数据为运算数。

分析：如果每次输入固定是三个数据，那就比较简单了，直接依次读入处理就可以得到这三个数据。但是这里给出的数据，可能是两个数字，也有可能是三个数字，所以就不能直接读入了。因此，可以将整条语句读进字符数组中，然后再根据字符串进行判断，根据不同情况分离出需要的数据，代码如下：

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
int main() {
    int n, a, b, c;
    char last, s[20], ans[20];
    scanf("%d\n", &n);
    while (n--) {
        fgets(s, sizeof(s), stdin); //读入一行
        if (s[0] == 'a' || s[0] == 'b' || s[0] == 'c')
            last = s[0], s[0] = ' '; //获取计算符号，并替换为空格
        sscanf(s, "%d %d", &a, &b); //从这个字符串里面读入两个数 a 和 b
        switch (last) {
        case'a': c=a+b; sprintf(ans, "%d+%d=%d", a, b, c); break; //+
        case'b': c=a-b; sprintf(ans, "%d-%d=%d", a, b, c); break; //-
        case'c': c=a*b; sprintf(ans, "%d*%d=%d", a, b, c); break; //x
        }
```

```
        printf("%s\n%d\n", ans, strlen(ans)); //输出
    }
}
```

本题使用了 fgets( ) 函数来进行读入一行字符串，并存入字符数组中，空格也一起存下了。之前常使用的 gets( ) 函数因为存在可能溢出的风险所以不使用。fgets(s,sizeof(s),stdin);这条语句中指定了字符数组的最大读入数量，因此是安全的。

接下来使用了 sscanf( ) 函数，可以从已经存储下来的字符串中读入信息。sprintf( ) 可以将信息输出到字符串中。回顾一下 scanf( ) 的用法，就会发现 sscanf( ) 和 scanf( ) 是很接近的。比如，sscanf(s,"%d",&a);就可以从 s 字符串中读入一个整数 a。它们的区别是，scanf( ) 是从标准输入中读入，而 sscanf( ) 是从给定的一个字符串中读入，所以要求提供字符数组的名称，表示从哪个字符串里面读入信息。

本题中的指令字符串第一个字符是 'a','b','c'，这会影响从这个字符串里面读入后面的信息，所以把这个字符赋值为空格，由于 scanf( ) 会自动忽视掉空格，所以这样可以规避这个问题。

同理 sprintf(s,"%d",a);就可以将一个 int 类型的数 a 输出到字符串 s 中而不是标准输出。请读者将这个函数和 printf( ) 进行比较。

6.2 string 类型字符串

使用 C 语言风格的字符数组有诸多不便，比如不能弹性变化长度，不能直接赋值或者复制，也有数组越界的风险。在 C++ 中提供了一些更好的工具——**标准模板库**(Standard Template Library，STL)，将很多有用的功能进行了封装，开箱即用，而不需要另外重新开发这些功能。STL 包括各类容器(如队列、栈等)、算法(如排序)和其他的一些功能。现在，将要使用 string 数据类型来处理字符串问题。

例 6-5　标题统计(洛谷 P5015，NOIP2018 普及组)。凯凯刚写了一篇美妙的作文，请统计这篇作文的标题中有多少个字符。注意：标题中可能包含大、小写英文字母、数字字符、空格和换行符，且字符串中的字符和空格数总和不超过 5。统计标题字符数时，空格和换行符不计算在内。

分析：因为读入字符串时会忽略前面的空格，并且读到分隔符(空格或者换行)时停止，所以可以一直读入字符串，每次读入一个字符串，把其长度加入答案中即可。代码如下：

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    int ans = 0;
    while (cin >> s)
        ans += s.length();
    cout << ans << endl;
    return 0;
}
```

这里没有使用字符数组，而是使用了一种新的“数据类型”——string。一个 string 类型的变量可以用来存储一个字符串，并且可以将这个字符串当作一个整体进行处理——可以对 string 进行赋值、拼接、裁切等，而字符数组毕竟是个数组，做到这些就比较麻烦了。

输入时使用 cin 语句，不断读入字符串。当发现读入文件读完后（遇到 EOF，在标准输入的时候可以按 Ctrl+Z 组合键），cin>>s 本身就会返回 0，中断 while 语句，结束读入。这里的 s 变量可以被认为是一个“加强版”的字符数组，可以使用 s.length( ) 来直接查询字符串 s 的长度，也可以和字符数组一样使用 s [0] 来查询这个字符串最开头的字符是什么。更厉害的是，string 类型的字符串可以直接拿来赋值、拼接操作，比如 s=s+s 就是将两个字符串 s 拼接在一起，其结果赋值回 s 的意思。这样操作的便利性可不是字符数组可以比拟的。

例 6-6　文字处理软件（洛谷 P5734）。现在需要开发一款文字处理软件。最开始时输入一个字符串（不超过 100 个字符）作为初始文档。可以认为文档开头是第 0 个字符，需要支持以下操作。

1）1 str：后接插入，在文档后面插入字符串 str，并输出文档的字符串。

2）2 a b：截取文档部分，只保留文档中从第 a 个字符起 b 个字符，并输出文档的字符串。

3）3 a str：插入片段，在文档中第 a 个字符前面插入字符串 str，并输出文档的字符串。

4）4 str：查找子串，查找字符串 str 在文档中最先出现的位置并输出；如果找不到输出 -1。

为了简化问题，规定初始的文档和每次操作中的 str 都不含有空格或换行。最多会有 q（$q \leqslant 100$）次操作。例如输入数据是：

```
4
ILove
1 Luogu
2 5 5
3 3 guGugu
4 gu
```

那么输出数据是：

```
ILoveLuogu
Luogu
LuoguGugugu
3
```

保证每次操作输入的字符串长度不超过 100 且输入合法（(2) 和 (3) 操作不会越界）。

分析：字符串 string 需要使用头文件 string，包括下面的常用方法。

1）string s：定义一个名字为 s 的字符串变量。

2）s+=str 或 s.append(str)：在字符串 s 后面拼接字符串 str。

3）s<str ：比较字符串 s 的字典序是否在字符串 str 的字典序之前。

4）s.size( ) 或 s.length( )：得到字符串 s 的长度。

5）s.substr(pos,len)：截取字符串 s，从第 pos 个位置开始 len 个字符，并返回这个字符串。

6）s.insert(pos,str)：在字符串 s 的第 pos 个字符之前，插入字符串 str，并返回这个字符串。

7) s.find(str, [pos]):在字符串 s 中从第 pos 个字符开始寻找 str,并返回位置,如果找不到返回 -1。pos 可以省略,默认值是 0。

对于本题来说,可以活用这些方法来处理字符串。

有一点需要注意的是,使用 find 函数查找子串但是找不到时,它会返回一个常数 string::npos,但是由于它不一定是个 int 类型的常量,所以需要将其强制转换为 int 类型才能直接输出 -1(读者可以试一下直接使用 cout 输出,这个数字会是什么)。

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    int n, opt, l, r;
    string s, a;
    cin >> n;
    cin >> s;
    while (n--) {
        cin >> opt;
        if (opt == 1) {
            cin >> a;
            s.append(a);
            // 使用 append() 函数，将 a 字符串加在 s 字符串后面
            cout << s << endl;
        } else if (opt == 2) {
            cin >> l >> r;
            s = s.substr(l, r);
            // 使用 substr() 函数，提取出 s 从 l 起的 r 个字符
            cout << s << endl;
        } else if (opt == 3) {
            cin >> l >> a;
            s.insert(l, a);
            // 使用 insert() 函数，将 a 字符串插入到 l 位置
            cout << s << endl;
        } else {
            cin >> a;
            cout << (int)s.find(a) << endl;
            // 使用 find() 函数，输出 a 字符串在 s 字符串中第一次出现的位置
        }
    }
    return 0;
}
```

string 类型就要简单很多了,可以直接赋值常量,也可以相互赋值。注意,字符数组不能直接像下面这样赋值:

```
string a, b;
a = "LUOGU";
b = a;
```

例 6-7 统计单词数(洛谷 P1308,NOIP2011 普及组)。给定一个单词(长度不超过 10,仅由英文字母组成),请输出它在给定的文章(长度不超过 1000000,包括字母和空格)中出现的次数和第一次出现的位置。注意:匹配单词时,不区分大小写,但要求完全匹配所有字母。

分析:由于匹配单词的时候不区分大小写,所以需要先将文章里面的单词和给定的这个单词中的大写字母统一转换为小写字母,然后再进行匹配。

还记得之前提到过的 find( )函数吗? 但是在这里使用 find( )函数会遇到这样一个问题:假设文本是 s = "i know that iakioi";a = "ak",进行 find(s,a) 会得到什么样的结果呢? 会得到 find(s,a) = 14,但是这篇文章中并没有一个单词是 "ak",而是 find( )函数把单词 iakioi 的一个子串 ak 认为是一个独立的单词了。

为了解决这个问题,将给定的单词前后都加一个空格,将其变成 " ak ",这样再使用 find( )函数,找出来的单词前后一定都有一个空格,所以不会找到一篇文章中单词的子串了(别忘了,文章前后也要加一个空格,应对需要的单词刚好在开头或者结尾的情况)。

那么,该如何统计这个给定单词在文章中的出现次数呢? 每次记录下 find( )函数返回的这个单词第一次出现的位置,如何从这个位置开始继续进行查找,这样就可以找出这个单词的出现次数了。

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string word, s;
    getline(cin, word);
    getline(cin, s);
    for (int i = 0; i < word.length(); i++)
        if ('A' <= word[i] && word[i] <= 'Z')
            word[i] += 'a' - 'A'; // 将给定单词的所有大写字母转换为小写字母
    for (int i = 0; i < s.length(); i++)
        if ('A' <= s[i] && s[i] <= 'Z')
            s[i] += 'a' - 'A'; // 将文章中的所有大写字母转换为小写字母

    word = ' ' + word + ' '; // 将给定单词前后都加上空格，防止多算
    s = ' ' + s + ' '; /* 文章的前后也需要加上空格，不然第一个和最后一个单词的统计可能出现问题 */

    if (s.find(word) == -1) {
        cout << -1 << endl;
    } else {
        int firstpos = s.find(word);
        int nextpos = s.find(word), cnt = 0;
```

```
        while (nextpos != -1) {
            cnt++;
            nextpos = s.find(word, nextpos + 1); /* 每次从上一次出现次数开始往后面查询这个单词下一次出现的位置 */
        }
        cout << cnt << " " << firstpos << endl;
    }
    return 0;
}
```

代码中使用了 getline( )函数,这可以将完整一行的输入数据读入到字符串中,无论这一行中是否有空格。使用方法是 getline(cin,字符串名称)。cin 是指输入流。

字符数组是 C 语言中就存在的,而在 C++ 中可以使用字符数组的“进化版本”,也就是 string。string 的变量名在很多情况下可以被当作字符数组的变量名,用于 sscanf、sprintf 等地方。string 和字符数组也是可以相互转换的,类似如下代码:

```
// string 转字符数组
char arr[10];
string s = "LUOGU";
int len = s.copy(arr, 9); // 最多允许复制 9 个字符，否则就越界了
arr[len] = '\0'; // 在末尾增加结束标记
// 或者
char arr[10];
string s = "LUOGU";
strcpy(arr, s.c_str()); // strncpy(arr, s.c_str(), 10);
// 字符数组转 string 就更简单了
char arr[10];
strcpy(arr, "LUOGU");
string s;
s = arr;
```

6.3 文件操作与重定向

到现在为止,输入输出方式都是标准输入输出(stdin,stdout):弹出一个窗口,手动输入内容,然后程序运行后,会在同一个窗口输出运行结果。包括洛谷在内,大多数 Online Judge 都是使用这种方式对程序进行评判的。

但是许多程序设计竞赛(比如 NOI 系列比赛)要求使用文件输入和输出。这种输出方式可以将硬盘上的文件调入程序,程序运算后生成另外一个文件。这一点非常重要,如果没有按照要求正确地使用文件输入输出,即使算法完全正确,也不能获得分数。

例 6-8　算法竞赛试题案例。以下是 NOIP2018 普及组复赛试题节选,请仔细阅读以下内容。

CCF 全国信息学奥林匹克联赛(NOIP2018)复赛

普及组

(请选手务必仔细阅读本页内容)

一、题目概况

中文题目名称	标题统计	龙虎斗	摆渡车	对称二叉树
英文题目与子目录名	title	fight	bus	tree
可执行文件名	title	fight	bus	tree
输入文件名	title.in	fight.in	bus.in	tree.in
输出文件名	title.out	fight.out	bus.out	tree.out
每个测试点时限	1s	1s	2s	1s
测试点数目	20	25	20	25
每个测试点分值	5	4	5	4
附加样例文件	有	有	有	有
结果比较方式	全文比较(过滤行末空格及文末回车)			
题目类型	传统	传统	传统	传统
运行内存上限	256MB	256MB	256MB	256MB

二、提交源程序文件名

对于 C++ 语言	title.cpp	fight.cpp	bus.cpp	tree.cpp
对于 C 语言	title.c	fight.c	bus.c	tree.c
对于 Pascal 语言	title.pas	fight.pas	bus.pas	tree.pas

三、编译命令(不包含任何优化开关)

对于 C++ 语言	g++ -o title title.cpp -lm	g++ -o fight fight.cpp -lm	g++ -o bus bus.cpp -lm	g++ -o tree tree.cpp -lm
对于 C 语言	gcc -o title title.c -lm	gcc -o fight fight.c -lm	gcc -o bus bus.c -lm	gcc -o tree tree.c -lm
对于 Pascal 语言	fpc title.pas	fpc fight.pas	fpc bus.pas	fpc tree.pas

注意事项:

1) 文件名(程序名和输入输出文件名)必须使用英文小写。

2) C/C++ 中函数 main( )的返回值类型必须是 int,程序正常结束时的返回值必须是 0。

3) 全国统一评测时采用的机器配置为:Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz,内存 32GB。上述时限以此配置为准。

4) 只提供 Linux 格式附加样例文件。

5) 特别提醒:评测在当前最新公布的 NOI Linux 下进行,各语言的编译器版本以其为准。

标题统计(title.cpp/c/pas)

【问题描述】

凯凯刚写了一篇美妙的作文,请问:这篇作文的标题中有多少个字符?

注意:标题中可能包含大、小写英文字母、数字字符、空格和换行符。统计标题字符数时,空格和换行符不计算在内。

【输入格式】

输入文件名为 title.in。

输入文件只有一行,一个字符串 s。

【输出格式】

输出文件名为 title.out。

输出文件只有一行,包含一个整数,即作文标题的字符数(不含空格和换行符)。

【输入输出样例 1】

title.in	title.out
234	3

见选手目录下的 title/title1.in 和 title/title1.ans。

【输入输出样例 1 说明】

标题中共有 3 个字符,这 3 个字符都是数字字符。

【输入输出样例 2】

title.in	title.out
Ca 45	4

见选手目录下的 title/title2.in 和 title/title2.ans。

【输入输出样例 2 说明】

标题中共有 5 个字符,包括 1 个大写英文字母,1 个小写英文字母和 2 个数字字符,还有 1 个空格。由于空格不计入结果中,故标题的有效字符数为 4 个。

【数据规模与约定】

规定 |s| 表示字符串 s 的长度(即字符串中的字符和空格数)。

对于 40% 的数据,$1 \leqslant |s| \leqslant 5$,保证输入为数字字符及行末换行符。

对于 80% 的数据,$1 \leqslant |s| \leqslant 5$,输入只可能包含大、小写英文字母、数字字符及行末换行符。

对于 100% 的数据,$1 \leqslant |s| \leqslant 5$,输入可能包含大、小写英文字母、数字字符、空格和行末换行符。

分析:本章刚刚分析了第一题“标题统计”的做法,写出了一段程序代码。但是选手在赛场上并不能直接提交写完的程序,必须要按照题目的要求提交代码。读者需要特别注意以下信息:

1) 提交源程序文件名,对于 C++,文件名为 title.cpp。

2) 输入文件名:title.in。

3) 输出文件名:title.out。

其余信息虽然也比较重要,但是现阶段还无法利用这些信息。按照前面的例题写完如下代码:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    int ans = 0;
    while (cin >> s)
        ans += s.length();
    cout << ans << endl;
    return 0;
}
```

根据要求,这个文件名应当保存为title.cpp,而不是未命名.cpp或者5015.cpp之类的文件名。然后需要将输入输出方式“重定向”到文件输入输出中。正确的方式如下:

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
int main() {
    freopen("title.in", "r", stdin);
    freopen("title.out", "w", stdout);
    string s;
    int ans = 0;
    while (cin >> s)
        ans += s.length();
    cout << ans << endl;
    return 0;
}
```

在主程序的开头多出了两条语句,同时还使用了cstdio头文件,这就是重定向输入输出到文件的方式,其一般形式为:

```
freopen("输入文件名", "r", stdin);
freopen("输出文件名", "w", stdout);
```

然后再尝试运行程序——一闪而过,但什么都没有!

别忘了,既然是文件输入输出,就应当给它输入文件啊!在title.cpp相同的文件夹里新建一个文件title.in,使用记事本打开这个文件,把样例输入“234”复制进去并保存[①]。这时,重新运行程序,还是一闪而过,但在文件夹里出现了一个title.out的文件。使用记事本打开这个文件,这就是这个程序的输出。

① 可能赛场上下发的压缩包里面有title1.in的样例输入文件,也可以把它复制到程序代码相同目录后重命名为title.in。由于系统差异,如果Windows下记事本不能正确显示换行,可以使用写字板打开。

但是，如果自己练习在 Online Judge 提交时，还要把重定向删除或者注释，以免无法通过评测。还有如下一种操作：

```
#ifndef ONLINE_JUDGE
    freopen("title.in", "r", stdin);
    freopen("title.out", "w", stdout);
#endif
```

在包括洛谷在内的许多 Online Judge 评测中，编译时会定义 ONLINE_JUDGE 宏。如果有检测到这个宏，就不会运行重定向操作，这样就可以在本地使用文件输入输出，在线提交使用标准输入输出了。

需要再次强调，代码文件名、输入输出必须完全和题目说明要求的一致，包括大小写、字母和数字等。保险起见，建议从赛题中直接复制文件名而不是重新手工输入一遍。

本地编写调试使用文件输入输出还有一个好处——不需要每次运行程序时都用键盘敲一遍输入。只需在记事本中写好输入文件并保存好，直接就可以运行程序了，这样节约时间。当然，还可以注释掉输出文件的重定向（第 7 行），仅保留输入重定向。这样运行程序结束时可以直接在弹出的窗口中观察到输出结果，进一步节约时间。

6.4 课后习题与实验

习题 6-1　手机（洛谷 P1765）。一般的手机键盘如图 6-3 所示。

1	2 abc	3 def
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz
*	0	#

图 6-3　手机键盘

要按出英文字母就必须要按数字键多下。例如，要按出 x 就得按 9 两下：第一下会出 w，而第二下会把 w 变成 x。0 键按一下会出一个空格。现在的任务是读取若干句只包含英文小写字母和空格的句子，求出要在手机上输入完这个句子至少需要按多少下键盘。

习题 6-2　honoka 的键盘（洛谷 P3741）。honoka 用一个只有 V 和 K 两个键的键盘输入了一个只有这两个字符的字符串。当这个字符串里含有“VK”这个字符串的时候，honoka 就特别喜欢这个字符串。所以，她想改变至多一个字符（或者不做任何改变）来最大化这个字符串内“VK”出现的次数。只有当“V”和“K”正好相邻时，则认为出现了“VK”。字符串的长度不超过 100。给出原来的字符串，请计算她最多能使这个字符串内出现多少次“VK”。

习题 6-3　单词覆盖还原（洛谷 P1321）。一个长度为 $l(3\leqslant l\leqslant 255)$ 的字符串中被反复贴有 boy 和 girl 两单词，后贴上的可能覆盖已贴上的单词（没有被覆盖的用句点表示），最终每个单词至少有一个字符没有被覆盖。问：贴有几个 boy 和几个 girl？

例如，输入的 ... boyogirlyy... girl... 是由 4 个 boy 和 2 个 girl 拼成的。

习题 6-4　数字反转升级版（洛谷 P1553）。给定一个数，请将该数各个位上的数字反转得到一个新数。

注意,本题与 NOIP2011 普及组第一题不同的是:这个数可以是小数、分数、百分数、整数。

1) 整数反转是将所有数位对调。

2) 小数反转是把整数部分的数反转,再将小数部分的数反转,不交换整数部分与小数部分。

3) 分数反转是把分母的数反转,再把分子的数反转,不交换分子与分母。

4) 百分数的分子一定是整数,百分数只改变数字部分。

整数新数也应满足整数的常见形式,即除非给定的原数为零,否则反转后得到的新数的最高位数字不应为 0;小数新数的末尾不为 0(除非小数部分除了 0 没有别的数,那么只保留 1 个 0);分数不约分,分子和分母都不是小数,本次没有负数。下面给出 4 个输入:

```
5087462
600.084
700/27
8670%
```

分别对应的输出是:

```
2647805
6.48
7/72
768%
```

习题 6-5　斯诺登的密码(洛谷 P1603)。给出一个含有 6 个单词的句子,需要破译密码,方法如下:

1) 找出句子中所有用英文表示的数字(≤20),列举如下:

正规:one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen twenty

非正规:a both another first second third

2) 将这些数字平方后对 100 取余,得到一个两位数。如果十位数是零,那么去掉。

3) 把这些数字按数位排成一行,组成一个新数。

4) 找出所有排列方法中最小的一个数,即为密码。

习题 6-6　你的飞碟在这儿(洛谷 P1200,USACO Training)。输入两行,每行是一个长度为 1~6 的字符串,每个字符串对应一个数字:将这个字符串中的每个字母对应的数字(A 对应 1,B 对应 2, ……Z 对应 26)相乘,然后对 47 取模。如果这两个字符串对应的数字是相同的,输出 GO,否则输出 STAY。

习题 6-7　语句解析(洛谷 P1597)。一串长度不超过 255 的 Pascal 语言代码,只有 a、b、c 3 个变量,而且只有赋值语句,赋值只能是一个一位的数字或一个变量,每条赋值语句的格式是[变量]:= [变量或一位整数];。未赋值的变量值为 0。输出 a、b、c 的值。

例如,当输入是“a:=3;b:=4;c:=5;”时,输出为“3 4 5”。

习题 6-8　(选做)垂直柱状图(洛谷 P1598)。编写一个程序从输入文件中去读取 4 行句子(全都是大写的,每行不超过 100 个字符),然后用柱状图输出每个字符在输入文件中出现的次数。严格地按照输出样例来安排你的输出格式。

请将程序命名为 diagram.cpp，并要求使用文件输入输出，输入文件是 diagram.in，输出文件是 diagram.out。请从洛谷对应的题面上复制数据，保存到 diagram.in 文件中作为输入文件，并检查输出文件是否和样例输出一致。

例如，当输入数据为：

```
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
THIS IS AN EXAMPLE TO TEST FOR YOUR
HISTOGRAM PROGRAM.
HELLO!
```

应当输出：

```
                            *
                            *
          *                 *
          *                 *       *   *
          *                 *       *   *
*         *       *         *       *   *
*         *       * *       * *     *   * * *
*         *     * * *       * *     * *   * * * *
*       * * * * * * *       * * * * * *   * * * *       * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```