字符和字符数组

在前面的学习中，我们重点使用了整型（int）和浮点型（double）数据类型，而且我们还刻意避开了一类基本类型，就是字符类型。

在计算机的数据处理中，字符类型是非常常见的，比如你在电脑上看到的这段文字，就属于字符类型。计算机是如何处理字符类型的呢？我们知道，计算机只能处理数字，确切来说只能处理 0 和 1 的数字，字符类型不属于数字，计算机也能处理吗？

这个我们需要先从 ASCII 表讲起。在计算机早期，科学家为了让计算机能够处理字符，特别制定了一个表，在这个表中规定了各个字符对应唯一的数字，这张表就叫做 ASCII（American Standard Code for Information Interchange，美国信息交换标准代码）表，ASCII 表制定了 256 个字符与其对应的数字，如下图：

ASCII表

（American Standard Code for Information Interchange 美国标准信息交换代码）

高四位 / 低四位		ASCII控制字符 0000 0						ASCII控制字符 0001 1						ASCII打印字符 0010 2		ASCII打印字符 0011 3		ASCII打印字符 0100 4		ASCII打印字符 0101 5		ASCII打印字符 0110 6		ASCII打印字符 0111 7		
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	‼	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	^Backspace 代码：DEL

例如，在这张表中，大写字母 A 对应的 ASCII 值就是 65，小写字母 a 对应的 ASCII 值就是 97。

通过这张表，我们知道了字符与数字的转换关系。当然，这里的字符没有表示汉字，表示汉字需要用到扩展的 ASCII 表，但是同理，每个汉字也都有与其对应的唯一数值，不独汉字，像日文、韩文、西班牙文等也一样，在这里就不展开了。

接下来，我们来看看 C++中如何处理字符。

字符变量

字符变量的定义：

char 字符变量;

例：char a='A';
char 关键字表示数据类型为字符，类似于 int、double、bool。

注意：
1、 char 类型只能存储一个字符，单个字符常量可以用单引号括起来，如'a'。
2、 字符类型是一个有序类型，字符的大小顺序按其 ASCⅡ码的大小而定。

例 1：输入一个字符，输出其对应的 ASCII 值

```
#include<iostream>
using namespace std;

int main(){
    char a;  //定义一个字符变量 a
    cin>>a;  //输入一个字符
    cout<<(int)a<<endl;  //输出字符对应的 ASCII 值。
}
```

例 2：输入一个整数（小于 255），输出其对应的字符。

```
#include<iostream>
using namespace std;

int main(){
    int a;
    cin>>a;
    cout<<(char)a; //（char）将 a 强制转换为 char 类型
}
```

例 3：按字母表顺序打印出 26 个大写字母，如下：
ABCDEFGHIJKLMNOPQRSTUVWXYZ

【分析】
因为字符类型是有序类型，字母的 ASCII 值是递增的，所以，我们可以结合循环遍历出 26 个字母，在输出时将对应的 ASCII 值转换成字符输出，详细看代码 1。另一种方法是我们可以直接对字符进行循环，直接输出字符循环变量，如下代码 2。

【代码 1】

```
#include<iostream>
using namespace std;
int main(){
    for(int i=0;i<26;i++){
```

```
        cout<<(char)('A'+i);
    }
}
```

【代码 2】

```
#include<iostream>
using namespace std;
int main(){
    for(char a='A';a<='Z';a++){
        cout<<a;
    }
}
```

字符的读入

在 C++里，字符可以用两种方式读入。

1、 使用 cin 读入，使用这种方式，无法读入空格、回车和制表符。
2、 使用 cin.get()读入可以读入任意字符。cin.get()可以不用参数，如：ch=cin.get()，也可以使用参数，如 cin.get(ch)，这两种使用方法是等价的，都把读入的字符赋值给字符变量 ch。

例 4：输入一个英文句子，统计句子中的字母个数。句子以‘#’号结束。
输入样例：
Thanks for being there, mom. Happy Mother’s Day.#
输出样例：
37
【分析】
在本题，我们需要不断读入字符，判断这个字符是否为字母，如果是，则计数器加一。
难点 1，怎么不断读入字符？可以通过 while 语句循环读入，直到这个字符为’#’则结束。
难点 2，怎么判断该字符是字母？字母在 ASCII 码中是一段连续的区间，只要该字符在这个区间内，就是字母，如以下代码。
【代码】

```
#include<iostream>
using namespace std;
char ch;
int cnt;
int main(){
    while(cin.get(ch)){  //这里的 cin.get(ch)也可以换成 cin>>ch
        if(ch=='#')break;
        if((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')){
            cnt++;
```

```
        }
    }
    cout<<cnt;
}
```

【思考】以上题目改成统计句子中的空格个数，应该怎么实现？

例 5：给出一段文本，将文本中的所有字母转换成大写后输出。
输入样例：
aaASDfasddsa@#$asd
asdfj@#%15415asfd512
as23d4as@#
输出样例：
AAASDFASDDSAASD
ASDFJASFD
ASDAS
【分析】
在本题中，不知有多少行，也不知每行有多少段，所以我们可以不断读入一个字符，直到文件结束。怎么判断读入结束呢？利用 cin.eof()可以返回流结束位，当 cin.eof()返回值为 true 则表示流读入结束，否则可以继续读入。以下程序可以实现不断读入文件，直到文件结束。

```
char gc;
while(!cin.eof()) {  //在控制台窗口中按 ctrl+Z 模拟文件结束
  cin>>gc;
  cout<<gc<<endl;
}
```

当然，以上程序也可以进一步优化，当使用 cin 或者 cin.get()函数时，都会返回值，当 cin 或者 cin.get()读不到数据时，就会返回流结束位，等价于 cin.eof()，所以以上程序可以优化为：

```
char gc;
while(cin>>gc){
  cout<<gc<<endl;
}
```

直到读入的方式后，就可以读取每个字符，对每个字符分类处理。特别需要注意的是，换行是一个字符，表示为'\n'。

【代码】

```
#include<iostream>
using namespace std;
char ch;
int main(){
```

```
    while(cin.get(ch)){
        if(ch>='a'&&ch<='z'){    //如果该字符是小写字母
            cout<<(char)(ch-'a'+'A');  //将小写转成大写
        }
        if(ch>='A'&&ch<='Z'){    //如果是大写字母
            cout<<ch;    //直接输出
        }
        if(ch=='\n')cout<<endl;  //如果是换行符则换行
    }
}
```

字符数组

字符数组定义格式跟一般数组相同，所不同的是数组类型是字符型，第一个元素同样是从 0 开始，而不是 1。具体格式如下：

char 数组名[常量表达式 1]…

例：
char ch1[5]; //数组 ch1 是一个具有 5 个字符元素的一维字符数组
char ch2[3][5]; //数组 ch2 是一个具有 15 个字符元素的二维字符数组

字符数组的初始化

1、用字符初始化数组
char chr1[5]={'a','b','c','d','e'};
当初始值个数少于元素个数时，从首元素开始赋值，剩余元素默认为空字符。
char chr2[5]={'a','b','c','d','\0'};
在一维字符数组中存放着带有结束符的若干个字符称为字符串。字符串是一维数组，但是一维字符数组不等于字符串。

2、用字符串初始化数组
char chr2[5]="abcd";
使用此格式均要注意字符串的长度应小于字符数组的大小或等于字符数组的大小减 1。

3、用字符串初始化二维字符数组
char chr3[3][4]={"abc","mno","xyz"};
在数组 ch3 中存放 3 个字符串，每个字符串的长度不得大于 3。

数组元素赋值

字符数组的赋值是给该字符数组的各个元素赋一个字符值。
char ch[3];
ch[0]='a';

ch[1]='b';
ch[2]='c';
对二维、三维字符数组也是如此。
当需要将一个数组的全部元素值赋予另一数组时，不可以用数组名直接赋值的方式，可以使用字符串拷贝函数 memcpy（）来完成。

字符常量和字符串常量的区别

（1）两者的定界符不同，字符常量由单引号括起来，如'A'，字符串常量由双引号括起来，如"ABC"。
（2）字符常量只能是单个字符，字符串常量则可以是多个字符。
（3)可以把一个字符常量赋给一个字符变量，但不能把一个字符串常量赋给一个字符变量。如 char ch；ch='A'，可以执行，ch="ABC"不可以执行，甚至 ch="A"也不可以执行。
（4）字符常量占一个字节，而字符串常量占用字节数等于字符串的字节数加 1。增加的一个字节中存放字符串结束标志“\0”。例如：字符常量'a'占一个字节，字符串常量"a"占二个字节。

字符数组的输入

字符数组的读入跟普通数组的读入不一样，可以直接读入整个字符串，而不需要逐个字符去读取。字符数组可以用以下两种方式读入：

1、cin

```
#include<iostream>
using namespace std;
char ch[100];
int main(){
    cin>>ch; //直接读入到字符数组
    cout<<ch;  //直接输出字符数组
}
```

在以上代码中，使用 cin 可以直接将字符串读入到 ch 字符数组中，也可以直接将字符数组输出，这个与普通数组有很大不一样。特别值得注意的是，使用 cin 读入，一旦遇到空格、回车、tab 符便停止读入，如上示例，输入"abc def"，ch 字符数组只读到了"abc"，空格及其后的字符停止了读入。

2、cin.getline()

如果要连空格也读入，我们可以使用 cin.getline()函数，cin.getline()函数需要两个参数，一个是读入的位置，另一个是读入的个数，如 cin.getline(ch,100)，指定读入到 ch 字符数组中，最多读入 100 个字符，如果提前遇到读入结束符如换行，则停止读入，否则一直读 100 个字符。其实，也可以利用第三个参数自定义读入结束符，如 cin.getline(ch,100,'#')，表示遇到'#'则读入结束。

```
1. #include<iostream>
2. using namespace std;
3. char ch[100];
4. int main(){
5.     cin.getline(ch,100);
6.     cout<<ch;
7. }
```

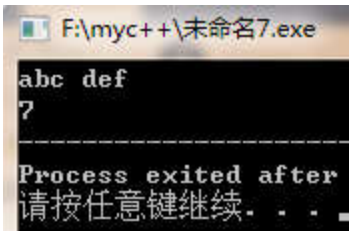

```
1. #include<iostream>
2. using namespace std;
3. char ch[100];
4. int main(){
5.     cin.getline(ch,100,'#');//自定义读入结束符
6.     cout<<ch;
7. }
```

字符数组函数

字符数组与普通数组不一样的地方还有，C++语言中为字符数组提供了非常丰富的函数，大大便利了字符数组的操作。在此我们介绍几个常用的字符数组函数，它们都需要共同的头文件 cstring。

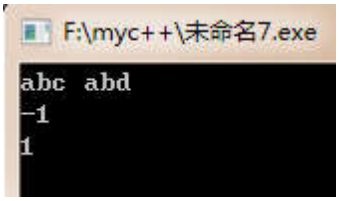
1、strlen(字符串名)
这个函数用来计算计算字符串的长度，终止符’\0’不算在长度之内

```
#include<iostream>
#include<cstring>
using namespace std;
char ch[100];
int main(){
    cin.getline(ch,100);
    cout<<strlen(ch);  //输出字符数组 ch 的长度
}
```



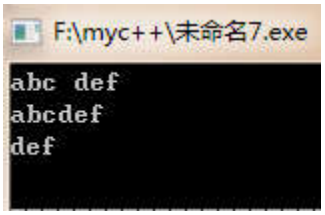
2、strcmp(字符串名 1,字符串名 2)
该函数比较字符串 1 和字符串 2 的大小，比较的结果由函数带回；
如果字符串 1>字符串 2，返回一个正整数 1；
如果字符串 1=字符串 2，返回 0；
如果字符串 1<字符串 2，返回一个负整数-1；
字符串比较的规则是：从左到右，按照字符的 ASCII 值逐位比较

```
#include<iostream>
#include<cstring>
using namespace std;
char a[100],b[100];
int main(){
    cin>>a>>b;
    cout<<strcmp(a,b)<<endl;
    cout<<strcmp(b,a)<<endl;
}
```



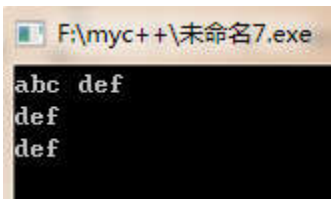
3、strcat(字符串名 1,字符串名 2)
该函数将字符串 2 连接到字符串 1 后边，返回字符串 1 的值。

```
#include<iostream>
#include<cstring>
using namespace std;
char a[100],b[100];
int main(){
    cin>>a>>b;
    strcat(a,b);
    cout<<a<<endl;
    cout<<b<<endl;
}
```



4、strcpy(字符串名 1,字符串名 2)
该函数将字符串 2 复制到字符串 1，返回字符串 1 的值。

```
#include<iostream>
#include<cstring>
using namespace std;
char a[100],b[100];
int main(){
    cin>>a>>b;
    strcpy(a,b);
    cout<<a<<endl;
    cout<<b<<endl;
}
```



例 6：输入 n 个单词，将每个单词转化成大写字母后按 ASCII 值从小到大排序输出。
输入样例：
4
Cat dog hello apple

输出样例：
APPLE
CAT
DOG
HELLO
【分析】
本题特别的地方在于比较的对象是字符数组。要存储 n 个单词，我们需要定义二维的字符数组，char a[100][100]，在这里，a[i]表示一个一维数组，可以用来存储一个单词。将一个字符数组转换成大写，可以直接用 strupr()函数，转成小写可以用 strlwr()函数。然后对 n 个字符数组进行排序比较。使用 strcmp()比较两个字符数组大小，使用 strcpy()可以交换两个字符数组。
【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
char a[100][100],ch[100];
int n;
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        strupr(a[i]); //将字符数组转换成大写，strlwr()可以转换成小写
    }
    //以下是选择排序
    for(int i=1;i<n;i++){
        for(int j=i+1;j<=n;j++){
            if(strcmp(a[i],a[j])>0){ //比较两个字符串大小
                //以下三行交换两个字符数组
                strcpy(ch,a[i]);
                strcpy(a[i],a[j]);
                strcpy(a[j],ch);
            }
        }
    }
    for(int i=1;i<=n;i++){
        cout<<a[i]<<endl;
    }
}
```

==========网站练习==========

字母统计

【题目描述】
晨晨刚上幼儿园，对字母很感兴趣，特别是对‘b’、‘B’、‘m’、‘M’四个字母感觉很亲切，因为这四个字母很像“爸”、“妈”的发音。每次看到一段英文文章，她都要数一数文章里面有多少个上面四个字母。由于她刚学数数，数不准，想让大哥哥、大姐姐帮她数一下，你能帮她吗？
【输入】
一行：输入一段以‘#‘结束的字符串。
【输出】
一行：一个整数代表字符串出现了多少个‘b’、‘B’、‘m’、‘M’字母。
【样例输入】
Thanks for being there, mom. Happy Mother’s Day.#
【样例输出】
4
【数据范围】
对于 80%的数据，字符串长度小于 255;
对于 100%的数据，字符串长度小于 1000;
【分析】
本题只需循环读入每个字符，对当前读入的字符，判断其是否为’b’、’B’、’m’、’M’，如果是则计数器加一。
【代码】

```
#include<iostream>
using namespace std;
char ch;
int cnt;
int main(){
    while(cin>>ch){
        if(ch=='#')break;
        if(ch=='b'||ch=='B'||ch=='m'||ch=='M')cnt++;
    }
    cout<<cnt;
}
```

手机

【题目描述】
一般的手机的键盘是这样的：

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

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。

你的任务是读取若干句只包含英文小写字母和空格的句子，求出要在手机上打出这个句子至少需要按多少下键盘。

【输入】

一行一个句子，只包含英文小写字母和空格，且不超过 200 个字符。

【输出】

一行一个整数，表示按键盘的总次数。

【样例输入】

i have a dream

【样例输出】

23

【分析】

基本思路，对读入的每个字符，判断该字符所需的按键次数，累加。但是对逐个字符进行判断代码难免冗长。我们可以定义一个常量数组，将各个字母所需的按键次数存储起来，如下代码第 3 行。其中 a[0]=1，表示‘a’的按键次数是 1。对于读入的字母 ch，它所对应的按键次数是 a[ch-‘a’]，ch-‘a’相当于将字母 ch 转换成对应的数值。如字母 b 的按键次数是 a[‘b’-‘a’]，即 a[1]，a[1]的值为 2。特别需要注意的是对空格的处理，本题需要读入空格，不能使用 cin 读入，应该使用 cin.get()读入。

【代码】

```
1.  #include<iostream>
2.  using namespace std;
3.  const int a[26]={1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,4,1,2,3,1,2,3,4};
4.  char ch;
5.  int sum;
6.  int main(){
7.      while(cin.get(ch)){
8.          if(ch==' ')sum++;
9.          if(ch>='a'&&ch<='z')sum+=a[ch-'a'];
10.     }
11.     cout<<sum;
12. }
```

ISBN 号码

【题目描述】

每一本正式出版的图书都有一个 ISBN 号码与之对应，ISBN 码包括 9 位数字、1 位识别码和 3 位分隔符，其规定格式如“x-xxx-xxxxx-x”，其中符号“-”就是分隔符（键盘上的减号），最后一位是识别码，例如 0-670-82162-4 就是一个标准的 ISBN 码。ISBN 码的首位数字表示书籍的出版语言，例如 0 代表英语；第一个分隔符“-”之后的三位数字代表出版社，例如 670 代表维京出版社；第二个分隔符后的五位数字代表该书在该出版社的编号；最后一位为识别码。

识别码的计算方法如下：

首位数字乘以 1 加上次位数字乘以 2……以此类推，用所得的结果 mod 11，所得的余数即为识别码，如果余数为 10，则识别码为大写字母 X。例如 ISBN 号码 0-670-82162-4 中的识别

码 4 是这样得到的：对 067082162 这 9 个数字，从左至右，分别乘以 1，2，…,9,再求和，即 0×1+6×2+……+2×9=158，然后取 158 mod 11 的结果 4 作为识别码。
你的任务是编写程序判断输入的 ISBN 号码中识别码是否正确，如果正确，则仅输出“Right”；如果错误，则输出你认为是正确的 ISBN 号码。
【输入】
输入只有一行，是一个字符序列，表示一本书的 ISBN 号码（保证输入符合 ISBN 号码的格式要求）。
【输出】
输出共一行，假如输入的 ISBN 号码的识别码正确，那么输出“Right”，否则，按照规定的格式，输出正确的 ISBN 号码（包括分隔符“-”）。
【样例输入】
输入样例 1
0-670-82162-4
输入样例 2
0-670-82162-0
【样例输出】
输出样例 1
Right
输出样例 2
0-670-82162-4
【分析】
模拟读入字符串，根据字符串的长度遍历每个字符，根据题目要求求出对应的权值，将权值跟最后一个字符比较。
【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
char isbn[15],check;
int sum,pow=1;
int main(){
    cin>>isbn;
    for(int i=0;i<strlen(isbn)-1;i++){
        if(isbn[i]!='-'){
            sum+=(isbn[i]-'0')*pow;
            pow++;
        }
    }
    sum%=11;    //求出权值
    if(sum==10)check='X';    //将权值转换成字符
    else check=sum+'0';
    if(check==isbn[12]){    //将权值和最后一位比较
        cout<<"Right";
    }else{
        isbn[12]=check;
```

```
        cout<<isbn;
    }
}
```

不明飞行物

【题目描述】
一颗彗星的后面有一个不明飞行物（UFO），这个 UFO 经常到地球上来寻找忠实的追随者，把他们带到宇宙中去。但由于舱内空间有限，它们每一趟只能带一组追随者。尽管如此，外星人仍然想出了一个妙法来决定带谁走：以 A 代表 1，B 代表 2，……Z 代表 26，USACO 即 21*19*1*3*15=17955，倘若此组人的组名所代表的数字与彗星的名字所代表的数字分别除以 47，余数相同，则彗星名与组名相匹配，UFO 带此组人飞向宇宙，余数不同则不匹配，故不带。
编程任务：
写一程序，打印出彗星名与组名是否相匹配，是打印"GO"，否打印"STAY"；同时打印出两者的余数。
【输入】
输入文件包含两行，第一行为慧星名，第二行为组名。
【输出】
由屏幕显示是否匹配的信息，下一行显示两者的余数。
【样例输入】
输入 1：
COMETHALEBOPP
HEAVENSGATE
输入 2：
SHOEMAKERLEVY
USACO
【样例输出】
输出 1：
GO
r1=r2=17
输出 2：
STAY
r1=21 r2=1
【分析】
本题训练点在于对字符数组的遍历。遍历每个字符，求出权值余数再进行比较。特别主要的是，数值乘积会很大，超过存储范围，需要边乘边模。
【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
char ch1[1000],ch2[1000];
int r1=1,r2=1;
```

```
int main(){
    cin>>ch1>>ch2;
    for(int i=0;i<strlen(ch1);i++){
        r1=r1*(ch1[i]-'A'+1)%47;
    }
    for(int i=0;i<strlen(ch2);i++){
        r2=r2*(ch2[i]-'A'+1)%47;
    }
    if(r1==r2){
        cout<<"GO"<<endl;
        cout<<"r1=r2="<<r1<<endl;
    }else{
        cout<<"STAY"<<endl;
        cout<<"r1="<<r1<<" r2="<<r2<<endl;
    }
}
```

笨小猴

【题目描述】
笨小猴的词汇量很小，所以每次做英语选择题的时候都很头疼。但是他找到了一种方法，经试验证明，用这种方法去选择选项的时候选对的几率非常大！
这种方法的具体描述如下：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，这样的单词很可能就是正确的答案。
【输入】
输入只有一行，是一个单词，其中只可能出现小写字母，并且长度小于 100。
【输出】
输出共两行，第一行是一个字符串，假设输入的的单词是 Lucky Word，那么输出“Lucky Word”，否则输出“No Answer”；
第二行是一个整数，如果输入单词是 Lucky Word，输出 maxn-minn 的值，否则输出 0。
【样例输入】
输入样例 1：
error
输入样例 2：
olympic
【样例输出】
输出样例 1：
Lucky Word
2
输出样例 2：
No Answer
0
【分析】

本题处理细节较多。

1、统计字母出现次数。可以类似桶排的方式记录每个字母出现的个数，在使用桶排方法时，注意将对应字母转换成数值。

2、统计最大数、最小数。遍历桶数组，可以统计出最大数最小数，要注意没出现的字符，即出现个数为 0 的字符不属于最小数。

3、判断素数。对最大数-最小数的值判断最小数。注意对 0 和 1 的特判。

【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
char word[1000];
int maxn,minn=1000,c[26];
int main(){
    cin>>word;
    for(int i=0;i<strlen(word);i++){
        c[word[i]-'a']++;  //桶排记录各字母出现的次数
    }
    for(int i=0;i<26;i++){
        if(c[i]==0)continue;  //出现次数 0 不在统计范围内
        if(c[i]>maxn)maxn=c[i];
        if(c[i]<minn)minn=c[i];
    }
    int n=maxn-minn;
    bool flag=true;
    if(n<2){    //判断素数，对 0 和 1 进行特判
        flag=false;
    }else{
        for(int i=2;i*i<=n;i++){
            if(n%i==0){
                flag=false;
                break;
            }
        }
    }
    if(flag==true){
        cout<<"Lucky Word"<<endl;
        cout<<n<<endl;
    }else{
        cout<<"No Answer"<<endl;
        cout<<0<<endl;
    }
}
```

扫雷

【题目描述】
你玩过扫雷吗？这个游戏在某个被淘汰的操作系统里有。游戏目标是找出 n*m 矩阵内的所有地雷。在本题中，你需要为每个单元格统计出它周围的地雷数。每个单元格最多和 8 个单元格相邻。左图矩阵有两个地雷，用“*”表示，计算结果如下图。

..*...****　　24*202****

...*　　***313****

.**..*.*..　　4**32*6*63

...*.**.　　*6*44*6**2

.***.*.*.*　　4***4*4*5*

***.*...*.　　***6*543*2

.*.****...　　3*6****543

..**..****　　23**55****

.*...*...*　　3*545*556*

.*.**.　　**3***3**2

【输入】
第一行有 n，m（0<=n，m<=100）,分别表示这个矩阵的行数与列数。接下来的 n 行每行包含 m 个字符“.”表示安全，“*”表示地雷。
【输出】
输出包含一个 n*m 矩阵。用数字表示该位置字符“.”周边的地雷数量，“*”还是表示地雷。
【样例输入】
4 4
*...
....
.*..
....
【样例输出】
*100
2210
1*10
1110

【分析】

本题看似二维字符数组，实际上我们可以把问题转换成数值，我们开一个二维数组，如果该点是雷，赋值为 1，如果不是雷，赋值为 0，统计该点的雷数，即是求其相邻八个点的数值和。怎样求相邻 8 个点的数值和呢？这里介绍一种坐标偏移的方法。假设当前点为(x,y)，则其相邻八个点的坐标分别为：

(x-1,y-1),(x-1,y+0),(x-1,y+1),(x+0,y+1),(x+1,y+1),(x+1,y+0),(x+1,y-1),(x+0,y-1)

我们提取里面的常量，存到数组中：

const int xx[8]={-1,-1,-1,0,1,1,1,0}; //坐标偏移数组，记录 x 轴方向

const int yy[8]={-1,0,1,1,1,0,-1,-1}; //坐标偏移数组，记录 y 轴方向

那么对(x,y)的第 i 个相邻点，我们可以直接用(x+xx[i],y+yy[i])获得。这样就可以直接利用循环来遍历(x,y)的八个相邻点了。

还有一个问题，有些点在边界上，它的相邻点越界了怎么办。我们将数组开大一圈，可以解决访问越界的问题。

【代码】

```
#include<iostream>
using namespace std;
char ch;
const int xx[8]={-1,-1,-1,0,1,1,1,0};  //坐标偏移数组，记录 x 轴方向
const int yy[8]={-1,0,1,1,1,0,-1,-1};  //坐标偏移数组，记录 y 轴方向
int a[102][102],n,m;
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>ch;
            if(ch=='*')a[i][j]=1;  //将数组转换成数值
            else a[i][j]=0;
        }
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            //枚举每个点，如果这个点是雷，直接输出‘*’
            if(a[i][j]==1){
                cout<<"*";
                continue;
            }
            //如果这个点不是雷，统计这个点周围的八个点，将雷数累加起来
            int cnt=0;
            for(int k=0;k<8;k++){
                cnt+=a[i+xx[k]][j+yy[k]];  //将相邻点的数值累加
            }
            cout<<cnt;
        }
        cout<<endl;
```

```
31.     }
32. }
```