实验舱
青少年编程
走近科学 走进名校

挑战信息学奥林匹克

C++程序设计（5）
字符串综合应用

string字符串使用小结

① string初始化

string s("abc123");

string s="abc123";

string s(10,'*';)

② 读取数据

cin

getline(cin, s)

③ “+”重载运算

字符串变量+字符串变量

字符串变量+字符串常量

字符串常量+字符串变量

例：

string s1, s2, s3;

s1 = “abc”;

s2 = s2 + ”123”;

s3 = “exp:” + s2;

cout << s3;

④ 可以使用关系运算符

>, >=, <, <=, ==, !=

例：

“abc” == “ABC” （false）

⑤ 数字与字符转换

'8’ - ‘0’

char(8 + ‘0’)

字符串的逆序

- 用循环

```
string s;
getline(cin, s);
int n = s.size();
for ( int i = n - 1; i >= 0; i-- ) cout << s[i];
cout << endl;
```

- 用函数

```
string s;
getline(cin, s);
reverse(s.begin(), s.end());
cout << s << endl;
```

例题-1：回文字符串

描述

一串字符如果从左读和从右读完全相同，我们称之为回文。

请判断键盘输入的一串字符（不超过 1000 位），判断是否是回文。

是，则输出 Yes，否则则输出 No。

输入：

```
abcba
```

输出：

```
Yes
```

参考代码

```
string s, st;
getline(cin, s);
st = s;
reverse(st.begin(), st.end());
if (s == st)
{
    cout << "Yes";
}
else cout << "No";
```

string类型的成员函数

函数	功能说明	举例说明
s.size()	获取字符串s的长度	s = “ab123”; cout << s.size(); //输出：5

s.find(st)	在字符串s中查找子串st，如果存在返回第一个子串位置，不存在则返回-1	string s = "ab cdefg";
		int p = s.find("cd");
		cout << p << endl; //输出：3

s.find(st, k)	从第k个位置开始查找子串st	
s.substr(k, len)	在字符串s中第k个位置开始取一个子串，子串长度len	string s = "ab cdef"; Cout << s.substr(3,2); //输出：cd
s. erase(k, len)	在字符串s中删除一个子串，k是子串开始的位置，len是子串长度	string s = "ab cdefg"; s.erase(2, 1); //结果：abcdefg
s.begin()	获取字符串s开始的位置	
s.end()	获取字符串s结束的位置	

字符串类型的成员函数

```
string s = "ABCDEFGH";
cout << s << endl;
int p = s.find("CD");
cout << p << endl;
p = s.find("xy");
cout << p << endl;
p = s.find("CD", 5);
cout << p << endl;
cout << s.substr(3, 2) << endl;
reverse(s.begin(), s.end());
cout << s <<endl;
```

```
ABCDEFGH
2
-1
-1
DE
HGFEDCBA
```

例题-2：搞破坏

某位大神闯进了实验舱的总部，打算搞一番破坏……他决定把桌面上的一份文件里面的每个单词都改一下。他要把这个单词的前面 k 个字母移到单词的末尾去。

输入

包含一个单词（只有英文字母，长度 ≤ 2000 ），一个正整数 k （小于单词长度）。

输出

输出被破坏后的单词。

样例输入　　样例输出

样例输入	样例输出
clubACM 4	ACMclub

参考代码

- 利用成员函数重新构造字符串

```
string s;
int k;
cin >> s >> k;
s += s.substr(0, k);
s.erase(0, k);
cout << s << endl;
```

```
clubACM
clubACMclub
ACMclub
```

例题-3:删除单词后缀

【描述】
给定一个单词，如果该单词以er、ly或者ing后缀结尾， 则删除该后缀（题目保证删除后缀后的单词长度不为0）， 否则不进行任何操作。
【输入】
输入一行，包含一个单词（单词中间没有空格，每个单词最大长度为32）。
【输出】
输出按照题目要求处理后的单词。
【样例输入】
referer
【样例输出】
refer

问题分析

样例：

Referer

Shopping

patiently

用substr()函数取出单词的后缀

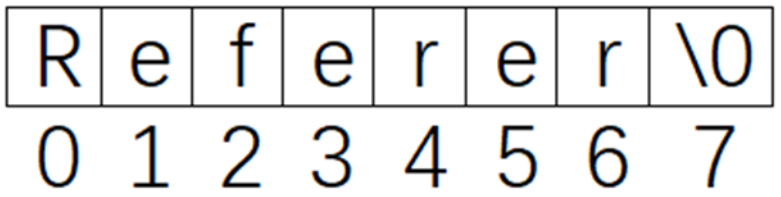
问题分析

样例：

Referer

Shopping

patiently



使用成员函数获取子串：

substr(k, len)

k：开始位置

k = s.size() - 2

k = s.size() - 3

s.substr(s.size() - 2, 2);

s.substr(s.size() - 3, 3);

参考代码

```
int len = s.size();
if (len > 3 && s.substr(len-3, 3) == "ing")
{
    s.erase(len - 3, 3);
}
else if (len >2 && (s.substr(len-2, 2) == "er"
     || s.substr(len-2, 2) == "ly"))
     {
        s.erase(len - 2, 2);
     }
cout << s << endl;
```

例题-4：回文子串

【描述】

给定一个字符串，输出所有长度至少为 2 的回文子串。

回文子串即从左往右输出和从右往左输出结果是一样的字符串，比如： $abba$ ， $cccdeedccc$ 都是回文字符串。

【输入】

一个字符串，由字母或数字组成。长度 500 以内。

【输出】

输出所有的回文子串，每个子串一行。

子串长度小的优先输出，若长度相等，则出现位置靠左的优先输出。

例题-4：回文子串

【样例输入】

12332112577516556 1

【样例输出】

33

11

77

55

2332

2112

5775

6556

123321

165561

算法分析

- 整行读取字符串s
- 用二重循环生成所有的子串：
  1. 外循环i从2~s.size()枚举所有子串的长度
  2. 内循环j从0~s.size() – i，枚举所有长度为i的子串，s.substr(j, i)
  3. 对于生成的子串判断是否回文字符串，如果是回文字符串，则输出该子串。

参考代码

```
int len = s.size();
for (int i = 2; i <= len; i++)
{
    for (int j = 0; j <= len - i; j++)
    {
        string s1 = s.substr(j, i);
        string s2 = s1;
        reverse(s2.begin(), s2.end());
        if ( s1 == s2 ) cout << s1 << endl;
    }
}
```

例题-5：循环移位

问题描述

Mas 喜欢字符串的循环移位。

通过移动字符可获得字符串的循环移位。从字符串的开头到字符串的结尾。例如，`ABCDE` 的循环移位为：`ABCDE`，`BCDEA`，`CDEAB`，`DEABC` 和 `EABCD`。
给定一些文本 T 和字符串 S，确定 T 是否包含 S 的循环移位。

输入格式

输入将完全由仅包含大写字母的两行组成。
第一行是文本 T，第二行是字符串 S。每行最多包含 1000 个字符。

输出格式

如果文本 T 包含字符串 S 的循环移位，则输出 yes。否则，输出 no。

输入样例1

```
ABCCDEABAA
ABCDE
```

输入样例2

```
ABCDDEBCAB
ABA
```

输出样例1

```
yes
```

输出样例2

```
no
```

算法分析

- 输入字符串t、s
- 循环生成s的所有循环移位字符串，并在t中进行查找；
  - 用s = s + s[0]; s.erase(0, 1);生成移位字符串；
  - 在t中查找s, t.find(s)
  - 如果能够查得到，则输出”yes”, 并退出程序；
- 如果循环正常结束，说明未查找到，则输出”no”。

参考代码

```
cin >> t >> s;
for (int i = 0; i < s.size(); i++)
{
    s += s[0];
    s.erase(0, 1);
    if (t.find(s) != -1)
    {
        cout << "yes";
        return 0;
    }
}
cout << "no";
```

练习讲解：ISBN号码

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 `-` 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；第二个分隔之后的五位数字代表该书在出版社的编号；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 $2\cdots$ 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 $1，2，\ldots，9$ ，再求和，即 $0\times1+6\times2+\cdots+2\times9=158$ ，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。

练习讲解：ISBN号码

【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】

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】

样例 #1：

```
0-670-82162-4
```

样例 #2：

```
0-670-82162-0
```

【样例输出】

样例 #1：

```
Right
```

样例 #2：

```
0-670-82162-4
```

算法分析

- 用cin读取ISBN码
- 用循环枚举字符串（最后一位字符不考虑），将数字字符转换为数字进行计算。

```
len = s.size();
for (int i = 0; i < len-1; i++)
{
    if (s[i] >= '0' && s[i] <= '9')
    {
        t++;
        n += t * (s[i] - '0');
    }
}
```

算法分析

- n对11求模 n = n % 11
- n与字符串最后一位数字字符进行比较，如果一致则输出。

```
if (n == (s[len-1] - '0') || s[len-1] == 'X' && n == 10)
{
    cout << "Right";
    return 0;
}
```

算法分析

- 如果计算的识别码与字符串最后一位数字字符不同，则替换该字符

```
if (n == 10)
{
    s[len-1] = 'X';
}
else s[len-1] = char(n + '0');
cout << s;
```

练习讲解：对称字符串

题目描述

Mas 认为对称的东西是美好的。于是他在黑板上写下了如下内容:

A_1 : A
A_2 : ABA
A_3 : ABACABA
A_4 : ABACABADABACABA

...

对于给定的 $N(1 \leq N \leq 20)$ 你能写出指定的 A_n 吗

输入格式

一个整数 n

输出格式

输出对应的 A_n

输入样例

```
2
```

输出样例

```
ABA
```

算法分析

1. 定义一个字符串变量s，赋初值”A”
2. 循环构造新的字符串：
   - 利用ASCII码生成第i个字符
   - 利用string类型变量可以相加的功能生成新的对称字符串
3. 输出字符串

```
for ( int i = 1; i < n; i++ )
{
    char c = 'A' + i;
    s = s + c + s;
}
```

练习讲解:找第一个只出现一次的字符

给出仅有小写字母的字符串，请找出第一个仅出现一次的字符。如果没有，输出 no

【输入格式】

一行字符串，长度小于 10000 。

【输出格式】

第一个仅出现一次的字符。如果没有，输出 no

【输入样例】

```
aacdhjrsss
```

【输出样例】

```
c
```

算法分析

1. 定义一个int类型的数组
2. 读取字符串s
3. 求取字符串长度len
4. 统计字符串中每个字符出现的次数（桶排）
5. 枚举字符串s，查找第一个出现一次的字母，输出并退出程序
6. 如果没有这样的字符输出“no”

算法分析

输出第一个仅出现一次的字符（查表法）

样例输入

akjahdkfacd

样例输出

j

```
for ( int i = 0; i < len; i++ )
{
    int x = s[i] - 'a';
    if (a[x] == 1)
    {
        cout << s[i] << endl;
        return 0;
    }
}
cout << "no" << endl;
```

a	b	c	d	e	f	g	h	i	j	k	……	z
3	0	1	1	0	1	0	1	0	1	2	0	0
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	……	a[25]

练习讲解：寻找最长单词

题目描述：

小明阅读英文句子时，想知道最长的单词，你能帮他找出来吗？

输入格式：

一行英文单词，每个单词用空格或逗号分隔，只包含字母、空格和逗号。单词由至少一个连续的字母构成，空格和逗号都是单词间的间隔。（数据确保只有一个最长单词）

输出格式：

第一行：最长单词，第二行：最长单词的长度。

【样例输入1】

```
I am studying Programming language C in Peking University
```

【样例输出1】

```
Programming
11
```

【样例输入2】

```
my te is abcdd,aaa
```

【样例输出2】

```
abcdd
5
```

算法分析

① 整行读取字符串；

② 求取字符串长度；

③ 枚举整个字符串，逐字符扫描：

- 如果字符不是空格或逗号，构成新单词（ss）；
- 如果字符是空格或逗号，说明一个单词被提取出来，检查单词长度是否最大值，如果是最大值，则记录单词长度nmax，记录单词smax
- ss清空

④ 枚举字符串结束后，再次检查最后一个ss是否最长单词

⑤ 输出最长单词和长度

参考代码

```
for (int i = 0; i < s.size(); i++)
{
    if ( s[i] != ' ' && s[i] != ',')
    {
        ss += s[i];
    }
    else {
        if ( ss.size() > nmax ){
            nmax = ss.size();
            smax = ss;
        }
        ss = "";
    }
}
```