字符串string

信智学院　陈琰宏

string串

string是C++标准库的一个重要的部分，主要用于字符串处理。可以使用输入输出流方式直接进行string操作，也可以通过文件等手段进行string操作。同时，C++的算法库对string类也有着很好的支持，并且string类还和c语言的字符串之间有着良好的接口。

一、初始化

初始化有两种方式，其中使用等号的是拷贝初始化，不使用等号的是直接初始化。

```
string str1 = "hello world";// str1 = "hello world"
string str2("hello world"); // str2 = "hello world"
string str3 = str1;      // str3 = "hello world"
string s(char c, char s_len) ;  // 将c字符串前chars_len个字符作为字符串s的初值。
string s(num, ‘c’) ;  // 生成一个字符串，包含num个c字符
```

二、获取长度（length、size）

```
string str = "hello world";
cout << str.length() << str.size();
```

```
string str = "hello world";
```

三、插入（insert）

```
str.insert(pos,n,ch) //在字符串s的pos位置上面插入n个字符ch
str.insert(6,4,'z'); // str = "hello zzzzworld"
str.insert(pos,str)  // 在字符串s的pos位置插入字符串str
str.insert(6,str2); // str = "hello hard world"
```

四、赋值（assign）

赋值也是一种初始化方法，与插入、替换、添加对应理解较为简单。

```
str.assign(n,ch)      将n个ch字符赋值给字符串s
str.assign(10,'h'); // str = "hhhhhhhhhh"
str.assign(s)    // 将字符串s赋值给字符串str
str.assign(temp); // str = "welcome to my blog"
```

五、删除（erase）

```
string str = "welcome to my blog";
s.erase(pos,n)    把字符串s从pos开始的n个字符删除
str.erase(11,3);// str = "welcome to blog"
```

六、取子串（substr）

```
string str = "The apple thinks apple is delicious";
s.substr(pos,n)   得到字符串s位置为pos后面的n个字符组成的串
string s1 = str.substr(4,5); // s1 = "apple"
s.substr(pos)     得到字符串s从pos到结尾的串
string s2 = str.substr(17);  // s2 = "apple is delicious"
```

七、比较（compare）

两个字符串自左向右逐个字符相比（按ASCII值大小相比较），直到出现不同的字符或遇’\0’为止。

若是遇到‘\0’结束比较，则长的子串大于短的子串，如：“9856” > “985”。如果两个字符串相等，那么返回0，调用对象大于参数返回1，小于返回-1。

```
string str1 = "small leaf";
string str2 = "big leaf";
```

s.compare(str)　比较当前字符串s和str的大小

```
cout << str1.compare(str2);     // 1
```

s.compare(pos,n,str)　比较当前字符串s从pos开始的n个字符与str的大小

```
cout << str1.compare(2,7,str2);// -1
```

s.compare(pos,n0,str,pos2,n)比较当前字符串s从pos开始的n0个字符与str中pos2开始的n个字符组成的字符串的大小

```
cout << str1.compare(6,4,str2,4,4);          // 0
```

八、交换（swap）交换两个字符串的值。

```
string str1 = "small leaf";
string str2 = "big leaf";
str1.swap(str2)  ,输出结果相同
swap(str1,str2);  // str1 = "big leaf"     str2 = "small leaf"
swap(str1[0],str1[1]);  // str1 = "ibg leaf"
```

九、反转（reverse） 反转字符串。

```
string str = "abcdefghijklmn";
reverse(str.begin(),str.end()); //str = "nmlkjihgfedcba"
```

十、 find函数

```
string str = "The apple thinks apple is delicious";
//长度34
string key = "apple";
```

s.find(str)　　查找字符串str在当前字符串s中第一次出现的位置

```
int pos1 = str.find(key);          // 4
```

s.find(str,pos)　　查找字符串str在当前字符串s的[pos,end]中第一次出现的位置

```
int pos2 = str.find(key, 10);      // 17
```

/从pos开始从后向前查找字符串s中前n个字符组成的字符串在当前串中的位置，成功返回所在位置，失败时返回string::npos的值

```
//从pos开始查找当前串中第一个在s的前n个字符组成的数组里的字符的位置。查找失败返回string::npos
int find_first_not_of(char c, int pos = 0) const;
int find_first_not_of(const char *s, int pos = 0) const;
int find_first_not_of(const char *s, int pos,int n) const;
int find_first_not_of(const string &s,int pos = 0) const;
//从当前串中查找第一个不在串s中的字符出现的位置，失败返回string::npos
int find_last_of(char c, int pos = npos) const;
int find_last_of(const char *s, int pos = npos) const;
int find_last_of(const char *s, int pos, int n = npos) const;
int find_last_of(const string &s,int pos = npos) const;
......
//find_last_of和find_last_not_of与find_first_of和find_first_not_of相似，只不过是从后向前查找
```

1 [1038]粗心的Tony

Tony是一个粗心的打字员，这不，他又犯错误了。更糟糕的是，光标键坏了，所以他只能用退格键回到出错的地方，纠正以后，对之后的正确字符又得重新输入，因为为了回到出错的字符，这些字符都删除了。

现在请你帮助他找到至少需要多长时间才能纠正错误。

输入文件的第一行是一个整数N，表示测试数据的个数。接下来有N个测试数据。每个测试数据占3行：

第1行是一个正整数t（≤100），表示Tony删除或者输入一个字符所花的时间。第2行是正确的文本内容。第3行是Tony输入的文本内容。

对每个测试数据，输出一行，为Tony纠正错误所花的最少时间。

1 [1038]粗心的Tony

输入文件的第一行是一个整数N，表示测试数据的个数。接下来有N个测试数据。每个测试数据占3行：

第1行是一个正整数t（≤100），表示Tony删除或者输入一个字符所花的时间。第2行是正确的文本内容。第3行是Tony输入的文本内容。

注意：文本只包含可读的字符，每行文本的字符数不超过80个。

对每个测试数据，输出一行，为Tony纠正错误所花的最少时间。

样例输入：

```
2
1
WishingBone
WashingBone
1
Oops
Oooops
```

样例输出：

```
20
6
```

分析

```
 5  int main()
 6  {
 7  int n, speed, i, j, l;
 8  string c,w;
 9  cin >> n;
10  for (i = 1; i <= n; i++)
11  {
12      cin >> speed;
13      getchar();
14      getline(cin, c);
15      getline(cin, w);
16      if (c.size() >= w.size()) { l = w.size(); }
17      else{ l = c.size(); }
18      for (j = 0; j < l; j++)
19      {
20          if (c[j] != w[j]) { break; }
21      }
22      printf("%d\n", (w.size() - j + c.size() -j)*speed );
23  }
```

2 [2118] 邮件的烦恼

最近四正的邮箱里有许多未读邮件，但是他又没有那么多的时间一一阅读，现在他只想阅读与acm有关的邮件，所以请你帮忙从邮箱中找出题目里带有“acm”（包括ACM或者acm。不包括Acm,aCm...）的邮件，输出需要查看的邮件数量。

输入格式

多组测试数据，每组测试数据第一行输入一个整数n，代表有n封邮件，接下来n行输入一行字符串，代表邮件的题目（长度不超过200）。

输出格式

输出需要查看的邮件数量。

样例输入：

2
acm TopCoder Special Round Match 600.5!
Amazon.cn
3
aa
Project Euler Email Validation System ACM
aacmm

样例输出：

1
2

分析

```
1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  int main()
5  {
6      string str, tmp;
7      int i, j, n, sum; int k = 0;
8      while (scanf("%d", &n) != EOF)
9      {
10         getchar();
11         sum = 0;
12         for (i = 1; i <= n; i++)
13         {
14             getline(cin, str);
15             for (j = 0; j < str.size() - 2; j++)
16             {
```

分析

```
1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  int main()
5  {
6      string str, tmp;
7      int i, j, n, sum; int k = 0;
8      while (scanf("%d", &n) != EOF)
9      {
10         getchar();
11         sum = 0;
12         for (i = 1; i <= n; i++)
13         {
14             getline(cin, str);
15             for (j = 0; j < str.size() - 2; j++)
16             {
```

```
13          {
14              getline(cin, str);
15              for (j = 0; j < str.size() - 2; j++)
16              {
17                  if (str.size() >= 3)
18                  {
19                      tmp = str.substr(j, 3);
20                      if (tmp == "acm" || tmp == "ACM")
21                      {
22                          sum++;
23                          break;
24                      }
25                  }
26              }
27          }
28          if (k == 0) { printf("%d", sum); }
29          else { printf("\n%d", sum); }
30          k++;
31      }
32  }
```

3 [1313] 垒沙漏图形

给定任意N个符号，不一定能正好组成一个沙漏。要求打印出的沙漏能用掉尽可能多的符号。

输入格式：

输入在一行给出1个正整数N（≤2000）。

输出格式：

首先打印出由*组成的最大的沙漏形状，最后在一行中输出剩下没用掉的星号数。

1 [1038]粗心的Tony

样例输入：

19

样例输出：

```
*****
 ***
  *
 ***
*****
2
```

分析

```
    int n, cnt = -1, remain;
    cin >> n;
    n++;
    while (n >= 0) {
        ++cnt;
        n -= (cnt * 2 + 1) * 2;
    }
    remain = n + (cnt * 2 + 1) * 2;
    --cnt;
    for (int i = cnt; i >= 0; --i)
    {
        string k(i * 2 + 1,'*');
        string blk(cnt - i,' ');
        cout << blk << k << endl;
    }
    for (int i = 1; i <= cnt; ++i)
    {
        string k(i * 2 + 1, '*');
        string blk(cnt - i, ' ');
        cout << blk << k << endl;
    }
    cout << remain;
```

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
int main()
{
    cin.tie(0); cout.tie(0);
    //cin，cout之所以效率低，是因为先把
    //要输出的东西存入缓冲区，再输出，
    //导致效率降低，而这段代码可以来
    //打消iostream的输入 输出缓存，
    //可以节省许多时间，使效率与scanf
    //与printf相差无几.
    int n, cnt = -1, remain;
    cin >> n;
```

今天的课程结束啦……

下课了…
同学们再见！