二.字符串

string

字符串

字符串：用双引号括起来的0个或多个字符。

比如 “Program = Datastructure + Algorithm"

字符串中的每一个符号都属于字符类型。简单的说多个字符连在一起构成了字符串。

字符串类型用关键字"string"来申明。

比如 string s="Hello world!!!";

那么就申明了一个字符类型的变量s,给它赋值的初值是"Hello world!!!"。

再比如 s="iphone?" 。那么就把变量s的值改为了"iphone?"

cout<<s; 输出的结果是 “iphone?”

输入方法cin>>s;

输出方法cout<<s;

注意：string是c++特有的，所以c语言的scanf()和printf()不能用于string类型

字符串的赋值

方法：

```
string s1="who r u?"
string s2=s1;
cout<<s2;
```

输出结果为“who r u?”

字符串的输入和输出只能用cin和cout：

```
cin>>s1;
cout<<s1;
```

字符数组的赋值

方法1：

```
char ch[20]="who r u?"
```

方法2：

```
char  ch[20]={'w','h','o',' ','r',' ','u','?'};
```

字符数组的输入和输出：

```
char ch[20];
```

可以用cin和cout：

```
cin>>ch;
cout<<ch;
```

也可以用scanf()和printf()：

```
scanf("%s",ch);
printf("%s",ch);
```

%s用于规定输入的是字符数组。

字符串的系统函数

```
string st1="To be ?";
string st2="or not to be ?"
```

1.求长度:

```
int a1=st1.length();
cout<<a1;
```

输出的结果是7。字符串st1中共有7个字符，包括空格。

2.连接两个字符串

```
string ss=st1+st2;
cout<<ss;
```

输出的结果是"To be ?or not to be ?"

"+"的作用是连接两个字符串

如果要在st1后面添加字符'!',

可以写成**st1=st1+"!";**

cout<<st1;结果就是To be ?!

字符数组的系统函数

```
char ch1[30]="To be ?";
char ch2[30]="or not to be ?";
```

1.求长度:

```
int a2=strlen(ch1);
```

cout<<a2; 输出的结果是7。字符数组ch1中共有7个字符，包括空格。

2.连接两个字符数组

```
strcat(ch1,ch2);
printf("%s",ch1);
```

输出的结果是"To be ?or not to be ?"

strcat(ch1,ch2)的作用是将ch1和ch2连接后再赋给ch1。

如果执行strncat(ch1,ch2,6);

表示把ch2的前6个字符连接在ch1之后，再赋给ch1。

```
printf("%s",ch)
```

输出的结果是"To be ?or not"

字符串与字符数组的比较都是从左往右一位一位对比字符的ASCII的大小。一旦在某一位分出胜负，结束比较。都是按字典序比较大小。

字符串的大小比较

```
string st1="ABCDEFGHIJK";
string st2="ABYD";
```

字符串的比较：
两个字符串可以直接用
>,<,>=,<=,==,!=来进行比较。

例如下列语句：

```
if(st1>st2)cout<<"big";
else if(st1<st2)cout<<"small";
        else if(st1==st2)cout<<"equal";
```

输出的结果是"small"

其原理是st1和st2从左往右一位一位的对比，当对比到st1[2]与st2[2]的时候，因为st1[2]的值是'C'，而st2[2]的值是'Y'，'Y'的ASCII码值要大于'C'的ASCII码值，所以得出结论st2要大于st1，比较结束。

字符数组的比较大小

```
char ch1[30]="APBCDEF";
char ch2[30]="APDA";
```

字符数组的比较：

```
int t=strcmp(ch1,ch2);
cout<<t;
```

输出的结果是-1

strcmp(ch1,ch2)用于比较两个字符数组，如果ch1==ch2相等则t的值为0，如果ch1<ch2那么t的值-1，如果ch1>ch2 那么t的值1

```
t=strncmp(ch1,ch2,2);
```

把ch1,ch2的前2个字符进行比较。

```
cout<<t;
```

输出的结果是0

```
string st1="To be ?";
string st2="or not to be ?"
```

3.字符串的替换

```
st1=st2;
cout<<st1;
```

输出的结果是"or not to be ?"

```
char ch1[30]="To be ?";
char ch2[30]="or not to be ?";
```

3.字符数组替换

```
strcpy(ch1,ch2);
printf("%s",ch1);
```

输出的结果是"or not to be ?"

strcpy(ch1,ch2)的作用是用ch2的值替换ch1的值。

如果执行strncpy(ch1,ch2,6);
表示用ch2的前6个字符替换ch1

```
printf("%s",ch1)
```

输出的结果是"or not"

字符串的其他函数

提取子串

```
string ss1="0123456789";
string ss2;
ss2=ss1.substr(pos,len); //将ss1中从pos开始的长度为len的子串赋给ss2;
 (例 ss2=ss1.substr (5,3) ; //此时ss2=="567") ;
```

子串查找

```
int w=ss1.find(ss2);
//查找ss2在ss1中出现的左起第一个位置，若找不到则返回-1;
(例 ss2="23"; int w=ss1.find(ss2);    //w==2
    ss2="00"; int w=ss1.find(ss2);    //w==-1);

int w=ss1.find(ss2,pos);
//从ss1的第pos(pos>=0)位开始查找，返回ss2第一次出现在ss1中的位置，若找不到则返回-1;
 (例 ss2="78"; int w=ss1.find(ss2,7) ;    //w==7
    ss2="78"; int w=ss1.find(ss2,8);     //w==-1) ;
```

字符串与字符数组 输入时需注意的问题

cin和scanf()都把空格和回车换行当成默认的输入间隔。
所以如果有

```
string st;
char ch[30];
```

如果要输入一句话"Hello NK!"到st和ch中，采用下列方式是不会成功的：

```
cin>>st;
```

cout<<st;输出的是"Hello"。
同理：

```
scanf("%s",ch);
```

printf("%s",ch);输出的也是"Hello"

原因都是系统吧Hello与NK!间的**空格当成是间隔符号**了。

如果要把**带空格的一行字符**读入一个字符串或一个字符数组中，可采用下列方式。
字符串：**getline(cin,st);**
字符数组：**gets(ch);**

字符串与字符数组 输入时需注意的问题

cin和scanf()都把空格和回车换行当成默认的输入间隔。
所以如果有
string st;
char ch[30];
如果要输入一句话"Hello NK!"到st和ch中，采用下列方式是不会成功的：

cin>>st;
cout<<st;输出的是"Hello"。
同理：
scanf("%s",ch);
printf("%s",ch);输出的也是"Hello"

原因都是系统吧Hello与NK!间的**空格当成是间隔符号**了。

如果要把**带空格的一行字符**读入一个字符串或一个字符数组中，可采用下列方式。
字符串：**getline(cin,st);**
字符数组：**scanf("%[^\n]%*c",&ch);**
可以写成 "%[^\n]%*c"，**%[^\n]的作用就是读\n**(\n表示回车换行)**之外的所有字符**，^表示不是，如果不是\n就继续读入。也就是说读到\n为止，%*c的作用就是把\n去掉，否则再次读的时候一直遇到的都是\n；