Cin>>:

1. 最常见的是获取输入的一个**字符**或**数字(后面会介绍只输入字符的情况),** cin>>会自动过滤掉不可见字符（如空格 回车 tab等）, 可以用noskipws流进行控制: cin >> noskipws >> input[i]
2. 获取输入的字符串，可以用数组或string类型。遇到空格、回车等会**结束获取输入的字符串**，后面的字符串会过滤掉（**存放在输入流中**）。如果后面还需要输入字符串，则会从前面存放的字符串开始获取。

**Cin**.get():

1. 此函数从输入流中读入一个**字符(char 非int，**如果定义数组或变量为int就会出错！**)**，返回值就是该字符的 ASCII 码。如果碰到输入的末尾，则返回值为 EOF。**get() 函数不会跳过空格、制表符、回车等特殊字符，所有的字符都能被读入。**
2. cin.get(字符变量名)，用来接收字符，只获取一个字符，可以接收空格，遇回车结束
3. cin.get(数组名，接收字符数目)，用来接收字符串，可以接收空格，遇回车结束。回车仍在输入流中

注意：数组的最后一个字符会是‘\0’，设接收字符数目为n，如果输入的字符串大于等于n，则实际接收到的输入是字符串的前面n-1个字符，包括空格（不包括回车，遇到回车就结束了），会自动在后面增加一个‘\0’。所以预设数组大小时要比实际输入大小大1.

1. cin.get()，没有参数，主要用于**舍弃**输入流中不需要的字符，或者舍弃回车，即舍弃输入流中的**一个字符(注意！是紧接着的一个字符！)**。

cin.getline():

第一个版本从输入流中读取 bufSize-1 个字符到缓冲区 buf，或遇到\n为止（**哪个条件先满足就按哪个执行**）。函数会自动在 buf 中读入数据的结尾添加\0。  
  
第二个版本和第一个版本的区别在于，第一个版本是读到\n为止，第二个版本是读到 delim 字符为止。\n或 delim 都不会被读入 buf，但会被从输入流中取走。  
  
这两个函数的返回值就是函数所作用的对象的引用。**如果输入流中\n或 delim 之前的字符个数达到或超过 bufSize，就会导致读入出错，其结果是：虽然本次读入已经完成，但是之后的读入都会失败。**  
  
从输入流中读入一行，可以用第一个版本。用cin >> str这种写法是不行的，因为此种读法在碰到行中的空格或制表符时就会停止，因此就不能保证 str 中读入的是整行。

总结来说就是，接收一个字符串，可以接收空格等，最后一个字符为‘\0’。结束符可以通过设置第三个参数自己设置，**默认是回车**。m不能为string类型。

**经常会遇到需要从输入流中读入一行的情况，可以用第一个版本，因为以回车符为结束标志。用cin >> str这种写法是不行的，因为此种读法在碰到行中的空格或制表符时就会停止，因此就不能保证 str 中读入的是整行。**

1. cin.getline(变量名， 输入大小)默认结束方式
2. cin.getline(变量名，输入大小，结束标志)自定义结束标志

getline():

用于string类的。使用需包含头文件#include<string>。getline(cin,string s)，**接收一个字符串**，可以接收空格、回车等

与cin.getline()的区别：

              1.cin.getline()接收输入字符串的是数组，getline（）是string类型。

              2.cin.getline()可以接收空格，但不能接收回车；getline()可以接收空格和回车

1. cin.getline()会在数组结尾是'\0'，getline()不会
2. 虽然getline也用了cin，但是他们存在区别：
3. cin与getline()的区别：
4. **getline()中的结束符，结束后，结束符不放入缓存区;**
5. **cin的结束符，结束后，结束符还在缓存区；**
6. 所以在使用 cin 后若要使用 getline() 必须要把前面cin遗留的结束符处理掉，解决方法为：在使用getline（）之前，加入一行getline（）来处理cin留下的结束符；

**gets()**

1.gets(m)用于string类的，需包含#include<string>。可以接收空格，遇回车结束。

是否需要使用cin.get()吃掉输入流中的回车要依据下一次读取的方式而定。如使用cin.getline()或类C输入scanf()或put()等的时候，就需要用cin.get()在下一次读取前清空残余缓存；若使用cin>>则不需要，因为cin是智能指针，它将跳过缓冲区的无效字符（如这里的回车），直到找到非空白字符为止，然后它将读取字符，直到再次遇到空白为止。