Using >>, cin.get, cin.getline, and cin.ignore

Using the >> operator (with cin)

The >> operator may be used when you simply want to read the next non-blankspace characters entered by the user into a character or character array. Any printable characters that follow the first space will be ignored and will not be stored in the variable. Do not use a statement like,

cin >> UserExplanation;

if, for example, you wish to obtain a whole sentence from the user that includes spaces. In that case, you would be better served by using the cin member functions get or getline.

Question: Will a null-terminator automatically be stored at the end of the character array, UserExplanation?

Using cin.get

The unformatted get member function works like the >> operator with two exceptions. First, the **get function includes white-space characters**, whereas the extractor excludes white space when the ios::skipws flag is set (the default). Second, the get function is less likely to cause a tied output stream (cout, for example) to be flushed.

A variation of the get function specifies a buffer address and the maximum number of characters to read. This is useful for limiting the number of characters sent to a specific variable, as this example shows:

```
#include <iostream.h>

void main()
{
   char line[25];
   cout << " Type a line terminated by carriage return\n>";
   cin.get( line, 25 );
   cout << ' ' << line;
}
```

In this example, you can type up to 24 characters and a terminating character. Any remaining characters can be extracted later.

Using cin.getline

The getline member function is similar to the get function. Both functions allow a third argument that specifies the terminating character for input. The default value is the newline character. **Both functions reserve one character for the required terminating character**. However, get leaves the terminating character in the stream and getline removes the terminating character.

The following example specifies a terminating character for the input stream:

```
#include <iostream.h>
```

```
void main()
{
   char line[100];
   cout << " Type a line terminated by 't'" << endl;
   cin.getline( line, 100, 't' );
   cout << line;
}
```

Using cin.ignore

cin.ignore(int nCount = 1, int delim = EOF);

Parameters

nCount - The maximum number of characters to extract.
delim - The delimiter character (defaults to EOF).

Remarks

Extracts and discards up to nCount characters. Extraction stops if the delimiter delim is extracted or the end of file is reached. If delim = EOF (the default), then only the end of file condition causes termination. The delimiter character is extracted.