Reading a string with spaces

Reading a string with spaces in C++ can be a bit tricky, as the standard **cin** function and the **getline()** function behave differently when it comes to handling whitespace characters. In this article, we'll discuss two common methods for reading strings with spaces in C++: using **getline()** and using **cin** with the **ignore( )** function.

Method 1: Using **getline( )** The **getline( )** function is specifically designed to read a line of text, including any whitespace characters that may be present. It takes two parameters: an input stream (such as **cin**) and a string variable to store the input. Here's an example of how to use **getline( )** to read a string with spaces:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string input;
    cout << "Enter a string with spaces: ";
    getline(cin, input);
    cout << "You entered: " << input << endl;
    return 0;
}
```

Method 2: Using **cin** with the **ignore()** function

Another method for reading strings with spaces is to use **cin** to read individual words, but then use the **ignore()** function to discard the remaining whitespace characters in the input stream. The **ignore()** function takes two parameters: an input stream and the number of characters to ignore. Here's an example of how to use **cin** and **ignore()** to read a string with spaces:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
```

```
    string input;
    cout << "Enter a string with spaces: ";
    cin >> input;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    getline(cin, input);
    cout << "You entered: " << input << endl;
    return 0;
}
```

Both of the above methods will allow you to read a string with spaces in C++. However, it's important to note that using the **cin** with **ignore()** will ignore the characters in the input stream until the given number of characters, specified in the second parameter of **ignore()** function, or until a newline character ('**\n**') is encountered.

When using **getline()**, you should be careful to remember that the newline character at the end of the input will be included in the string, which may cause problems if you're expecting the string not to contain a newline character.

```
Note that we have only discussed Method 1 as it is the one
which is widely used and
accepted.
```

It's also worth noting that if you expect multiple lines of input to be passed, it's better to use **getline()** as it's designed to read entire lines of input, whereas **cin** is designed to read individual words.

In conclusion, both method can be used for reading strings with spaces in C++ but the appropriate method should be chosen according to the requirement of the problem.