

```
1 #include "Word.h"
2
3 Word::Word()
4 : uintCount( 1 )
5 {
6     /* empty */
7 }
8
9 Word::Word( string strWord )
10 : strWord( strWord ), uintCount( 1 )
11 {
12     /* empty */
13 }
14
15 Word::~Word()
16 {
17     /* empty */
18 }
19
20 ostream& operator << ( ostream& objostreamOut, const Word*& objWordToOutput )
21 {
22     unsigned int uintWordLength = objWordToOutput->strWord.length();
23
24     for( unsigned int uintI; uintI < uintWordLength; uintI++ )
25     {
26         objostreamOut << toupper( objWordToOutput->strWord.at( uintI ) );
27     }
28
29     objostreamOut << " " << objWordToOutput->getCount();
30
31     return objostreamOut;
32 }
33
34 bool Word::operator == ( Word& objWordCompare )
35 {
36     unsigned int uintWordLength      = this->strWord.length();
37     string strCompareValue          = objWordCompare.getValue();
38
39     if( uintWordLength != strCompareValue.length() )
40     {
41         return false;
42     }
43     else
44     {
45         for( unsigned int uintI = 0; uintI < uintWordLength; uintI++ )
46         {
47             if( toupper( this->strWord.at( uintI ) ) != toupper( strCompareValue.at( uintI ) ) )
48             {
49                 return false;
50             }
51         }
52     }
53 }
```

```
51     }
52 
53     return true;
54 }
55 }
56
57 bool Word::operator < ( Word& objWordCompare )
58 {
59     unsigned int uintSmallestWordLength = 0;
60     string strCompareValue           = objWordCompare.getWord();
61
62     if( this->strWord.length() <= strCompareValue.length() )
63     {
64         uintSmallestWordLength = this->strWord.length();
65     }
66     else
67     {
68         uintSmallestWordLength = strCompareValue.length();
69     }
70
71     for( unsigned int uintI = 0; uintI < uintSmallestWordLength; uintI++ )
72     {
73         if( toupper( this->strWord.at( uintI ) ) < toupper( strCompareValue.at( uintI ) ) )
74         {
75             return true;
76         }
77         else if( toupper( this->strWord.at( uintI ) ) > toupper( strCompareValue.at( uintI ) ) )
78         {
79             return false;
80         }
81     }
82 }
83
84 bool Word::operator > ( Word& objWordCompare )
85 {
86     unsigned int uintSmallestWordLength = 0;
87     string strCompareValue           = objWordCompare.getWord();
88
89     if( this->strWord.length() <= strCompareValue.length() )
90     {
91         uintSmallestWordLength = this->strWord.length();
92     }
93     else
94     {
95         uintSmallestWordLength = strCompareValue.length();
96     }
97
98     for( unsigned int uintI = 0; uintI < uintSmallestWordLength; uintI++ )
99     {
100        if( toupper( this->strWord.at( uintI ) ) > toupper( strCompareValue.at( uintI ) ) )
```

```
101         {
102             return true;
103         }
104     else if( toupper( this->strWord.at( uintI ) ) < toupper( strCompareValue.at( uintI ) ) )
105     {
106         return false;
107     }
108 }
109
110 string Word::getWord() const
111 {
112     return this->strWord;
113 }
114
115 string Word::getValue() const
116 {
117     return this->strWord;
118 }
119
120
121 void Word::setWord( string strWord )
122 {
123     this->strWord    = strWord;
124 }
125
126 unsigned int Word::getCount() const
127 {
128     return this->uintCount;
129 }
130
131 void Word::incrementCount()
132 {
133     this->uintCount++;
134 }
135
```