

Laboratory Exercise 6 (Operator Overloading)

Topics: overloaded operators in a class definition

Goals: Upon successful completion of this lab you should be able to:

- Declare and implement class operators

- Use class operators in an application program

- Sort an array of class objects

- Describe the relationship between the constructor, copy constructor, destructor, and assignment operator for a class

Related text sections:

- Chapter 7

- Chapter 8

Laboratory Exercise 6 Instructions

1. Copy your header file for lab 5 to a file called Lab6.h.
2. Remove the prototype and implementation for “WriteData” from Lab6.h.
3. Add the following prototypes to the public section of your WordData class description:

```
WordData & operator = (const WordData & OtherWord);  
bool operator == (const WordData & OtherWord) const;  
WordData operator + (const WordData & OtherWord) const;  
friend istream & operator >> (istream & ins, WordData & w);  
friend ostream & operator << (ostream & outs, const WordData &  
w);
```
4. Implement the assignment operator for the WordData class:

```
WordData & operator = (const WordData & OtherWord);
```

Specifications:
 - a. Will check for self assignment.
 - b. Will release dynamically allocated space.
 - c. Will dynamically allocate new space.
 - d. Will copy values from OtherWord to this WordData object.
 - e. Will return this.
5. Implement the equality operator for the WordData class:

```
bool operator == (const WordData & OtherWord) const;
```

Specifications:
 - a. Will check for self testing.
 - b. Will return true if this WordData object is identical to OtherWord.
 - c. Will return false if this WordData object is not identical to OtherWord
6. Implement the addition operator for the WordData class:

```
WordData operator + (const WordData & OtherWord) const;
```

Specifications:
 - a. Will create a new WordData object.
 - b. Will concatenate OtherWord to the end of this WordData object.
 - c. Will add count values
 - d. Will return new WordData object.
7. Implement the input operator for the WordData class:

```
friend istream & operator >> (istream & ins, WordData & w);
```

Specifications:
 - a. Will read in a new word from the input stream (ins).
 - b. Will return if the input stream (ins) fails.
 - c. Will store the word read from ins into w.
 - d. Will calculate the counts.
 - e. Will return ins.
8. Implement the output operator for the WordData class:

```
friend ostream & operator << (ostream & outs, const WordData &  
w);
```

Specifications:

- a. Will write the value of word and the value of each of the counts on a single line separated by tabs. Will not write a new line.
 - b. Will return outs.
9. Copy your program file Lab5b.cpp to a file called Lab6a.cpp.
10. Remove all references to WriteData and SetWord from your Lab6a program.
11. Modify your Lab6a program to test all of the new methods you added to the class.
12. When you are satisfied that your class is working correctly, drop your modified header file in the ~tiawatts/cs215drop folder as *yourlastnameL6.h*.
13. Ask me to test your class.

