

Laboratory Exercise 4 (C++ classes)

Topics: C++ classes

Class constructors

Class destructor

Class accessor functions

Class mutator functions

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

1. Write a description of a simple C++ class
2. Implement the member functions of a C++ class
3. Describe the purpose of a class constructor
4. Describe the purpose of a class destructor

Related text sections:

Chapter 6

Laboratory Exercise 4 Instructions

1. Copy the file Lab4.cpp from the pickup folder:
cp ~tiawatts/cs215pickup/Lab4.cpp .

```
// Title: Lab 4 Driver program
// Author: Dr. Watts
// Description: This program is designed to test the WordData class.

#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include "Lab4.h"

using namespace std;

const int MAX = 100;

int main ()
{
    // Open file for input
    ifstream input ("words.txt");
    string inword;
    WordData words[MAX];
    // Initialize word counter
    int count = 0;
    // Read until array is filled or end of file is found
    while (count < MAX && input >> inword)
    {
        // Insert word into array
        words[count].SetWord(inword);
        // Increment counter
        count ++;
    }
    // Close input file
    input.close();
    // Print report header
    cout << setw (12) << left << "Word";
    cout << setw (8) << right << "Vowels";
    cout << setw (8) << right << "Const.";
    cout << setw (8) << right << "Digits";
    cout << setw (8) << right << "Special";
    cout << endl;
    // Loop through all words in array
    for (int i = 0; i < count; i++)
    {
        // Print data for word
        words[i].WriteData(cout);
        cout << endl;
    }
    return 0;
}
```

- Using a text editor, enter the following preprocessor directives and class description into a file called Lab4.h:

```
#ifndef WORDDATA
#define WORDDATA

#include <cstring>
#include <string>
#include <iostream>
#include <iomanip>

using namespace std;

class WordData
{
public:
    WordData ();
    WordData (const WordData & OtherWord);
    ~WordData ();
    void SetWord (const string & InWord);
    string GetWord () const;
    void WriteData (ostream & outs) const;
private:
    char * word;           // C-string to hold the word
    int vowels;           // vowel counter
    int consonants;       // consonant counter
    int digits;           // digit counter
    int specialchars;     // special character counter
};

// Function implementations will be placed here

#endif
```

- Try to compile the file Lab4.cpp.
`g++ Lab4.cpp -o lab4`
The linker should return errors because only the function prototypes have been declared; the functions must still be implemented.
- Implement the default constructor: `WordData ()`;
Specifications:
 - Will set the word to `new char [1]`; with `word[0] = '\0'`;
 - Will set the counters to 0.
- Implement the copy constructor:
`WordData (WordData & OtherWord)`;
Specifications:
 - Will copy the word from `OtherWord`
 - Will copy the counters from `OtherWord`

6. Implement the destructor: `~WordData ()`;

Specifications:

- a. Will release the memory used to store word

7. Implement the mutator function:

```
void SetWord (string InWord);
```

Specifications:

- a. Will allocate space to store the C string associated with InWord
- b. Will count the vowels in InWord (a, e, i, o, or u)
- c. Will count the consonants in InWord (letters that are not vowels)
- d. Will count the digits in InWord (0, 1, 2, 3, 4, 5, 6, 7, 8, or 9)
- e. Will count the “special characters” in InWord (characters that do not fall in any of the other categories)

8. Implement **and test** the accessor function: `string GetWord ()`;

Specifications:

- a. Will return the C string word as a C++ string

9. Implement the output function:

```
void WriteData (ostream & outs);
```

Specifications:

- a. Will write the value of word, left justified in 12 spaces.
- b. Will write the counter values, each right justified in 8 spaces.
- c. No new line at the end.

10. Compile your program:

```
g++ Lab4.cpp -o lab4
```

11. Execute your compiled program. The output should be:

Word	Vowels	Const.	Digits	Special
This	1	3	0	0
file	2	2	0	0
contains	3	5	0	0
15	0	0	2	0
words.	1	4	0	1
It	1	1	0	0
has	1	2	0	0
lots	1	3	0	0
of	1	1	0	0
letters	2	5	0	0
and	1	2	0	0
very	1	3	0	0
few	1	2	0	0
special	3	4	0	0
characters!	3	7	0	1

12. When your program is executing correctly, drop your completed header file in the `~tiawatts/cs215drop` folder as `yourlastnameL4.h`.