Lab 6: Extra Credit Problems

Thomas Bischoff

Original

Date: 4 / 3 / 2018

Problem 1: unique\_word\_finder.cpp

The function of this program is to find unique words within a string which begin with a capital letter and is neither ‘I’ or the first letter of the sentence.

Pseudocode 1: unique\_word\_finder.cpp

int main() –

1. Initialize a variable to hold a string provided by the user of the program.
2. Run the function string user\_provided\_string() and have the function initialize in step one hold it.
3. Initialize a variable and set it equal to the length of the user’s string.
4. Initialize a variable to hold the number of words contained in the user’s string.
5. Run the function int word\_counter(string str, int size); where str is the user’s string and size is the length of that string; and set the value it returns to be held by the variable initialized in step four.
6. Initialize an array to hold each word in the string with a length equal to the value returned in step five.
7. Initialize a variable for the starting place of the array and set it equal to ‘0’ to begin with.
8. Go through the entirety of the user’s provided string.

A. Check if the current character in the string is a 000space.

i. If the character is a space do the following:

- Increase the variable initialized in step seven 00000000by one.

ii. If the character is not a space do the following:

- Add the character into the current word in the 00000000array initialized in step six.

1. Display a title introducing the list of unique words in the string.
2. Run the function void unique\_words(string \*ptr\_array, int length, int start); where ptr\_array is the array created in step six, length is the value held in the variable initialized in step four, and start is equal to ‘1’.

Pseudocode 1: unique\_word\_finder.cpp

string user\_provided\_string() –

1. Initialize a variable to take the string that the user will input.
2. Ask the user to input a string into the program.
3. Take the input which the user has provided and set it to be held by the variable initialized in step one.
4. Return the variable containing the user’s string.

int word\_counter(string str, int size) –

1. Initialize a variable for the word count and set it equal to ‘1’.
2. Go through the string; where the string is str.

A. Check if the current character is a space

i. If the character is a space do the following:

- Add one to the variable initialized in step one.

1. Return the variable initialized in step one.

void unique\_words(string \*ptr\_array, int length, int start) –

1. Check if value for start is less than the value for length.

A. Go through each element contained in the array; where 000the array is ptr\_array.

i. Check if the element which is currently be viewed 000000is already in the array.

- If it is already in the array do the following:

+ Re-run the function void unique\_words(string 0000000000\*ptr\_array, int length, int start); where the 0000000000value at start has been increased by one.

B. Set a variable to hold the first character of the 000element within the array; where the array is 000ptr\_array.

C. Check to make sure that the variable initialized 000in step B is not ‘I’.

i. If it’s not ‘I’ do the following:

- Initialize a variable and set it equal to the 00000000ASCII number that corresponds to the character 00000000held in the variable initialized in step B.

- Check to see if the variable’s ASCII value is 00000000greater than 64.

+ If it is do the following:

a. Check if the variable’s ASCII value is less 0000000000000than 91.

Pseudocode 1: unique\_word\_finder.cpp

void unique\_words(string \*ptr\_array, int length, int start) –

= If it is less than 91 do the following:

~ Display the word to the user.

D. Re-run the program void unique\_words(string 000\*ptr\_array, int length, int start; where start has 000been increased by one.

Code 1: unique\_word\_finder.cpp

1. //######################################################################
2. // Program Header: unique\_word\_finder.cpp
3. // The Function of This Program is to Find Unique Words Within a String
4. // Which Start With a Capitial and is Neither an 'I' or the Start of the
5. // Sentence.
6. // Author: Thomas Bischoff
7. // Date Created: 3/27/2018
8. //######################################################################
10. #include <iostream>
11. #include <string>
12. #include <cctype>
14. **using** **namespace** std;
16. // Function Declarations
17. string user\_provided\_string();
18. **int** word\_counter(string str, **int** size);
19. string \*word\_collector(string str, **int** length, **int** word\_amount);
20. **void** unique\_words(string \*ptr\_array, **int** length, **int** start);
22. **int** main()
23. {
24. // Initialize a Variable to Hold the User's Provided String
25. string users\_string;
26. // Set users\_string to Hold the String Returned by the Function
27. users\_string = user\_provided\_string();
28. // Intialize a Variable for the Length of the String
29. **int** string\_length = users\_string.length();
30. // Initialize a Variable for the Number of Words
31. **int** amount\_of\_words;
32. // Set amount\_of\_words to Hold the Integer Returned by the Function
33. amount\_of\_words = word\_counter(users\_string, string\_length);
34. // Initilaize an Array for the Words in the String
35. string word\_array[amount\_of\_words];
36. // Initialize a Variable for the Starting Place of the Array
37. **int** starting = 0;
38. // Go Through the User's String
39. **for** (**int** i = 0; i < string\_length; i++)
40. {
41. // Check if the Current Character in the String is a Space
42. **if** (users\_string[i] == ' ')
43. {
44. // Add one to starting
45. starting += 1;
46. }
47. // Otherwise
48. **else**
49. {
50. // Add the Character to the Current Word in the Array
51. word\_array[starting] += users\_string[i];
52. }
53. }
54. // Display the Unique Words From the String
55. cout << "Unique Words:" << endl;
56. // Run the Function
57. unique\_words(word\_array, amount\_of\_words, 1);
58. }
60. // Function Definitions
61. string user\_provided\_string()
62. {
63. // Initialize a Value to Take the User's Input
64. std::string users\_input;
65. // Ask the User to Input a String into the Program
66. std::cout << "Please Enter a String into the Program: ";
67. // Take the Input and Set it Equal to users\_input
68. std::getline(std::cin, users\_input);
69. // Return users\_input
70. **return** users\_input;
71. }
73. // Function Definition
74. **int** word\_counter(string str, **int** size)
75. {
76. // Intialize the a Variable for the Word Count
77. **int** word\_count = 1;
78. // Go Through the Provided String
79. **for** (**int** i = 0; i < size; i++)
80. {
81. // Check if the Current Character is a Space
82. **if** (str[i] == ' ')
83. {
84. // Add One to word\_count
85. word\_count += 1;
86. }
87. }
88. // Return word\_count
89. **return** word\_count;
90. }
92. // Function Definition
93. **void** unique\_words(string \*ptr\_array, **int** length, **int** start)
94. {
95. // Check if start is Less Than length
96. **if** (start < length)
97. {
98. // Go Through the List of Words
99. **for** (**int** i = start + 1; i < length; i++)
100. {
101. // Check If the Word Repeats in List
102. **if** (ptr\_array[start] == ptr\_array[i])
103. {
104. // Re-run the Function Increasing Start by One
105. unique\_words(ptr\_array, length, (start + 1));
106. }
107. }
108. // Set a Variable to Hold the First Character in the Word
109. **char** first\_letter = ptr\_array[start][0];
110. // Check to see That it is not 'I'
111. **if** (!(first\_letter == 'I'))
112. {
113. // Intialize a Variable for ASCII Value for the Character
114. **int** ascii\_value = first\_letter;
115. // Check if the Letter is Uppercase
116. // Check if ascii\_value is Greater Than 64
117. **if** (ascii\_value > 64)
118. {
119. // Check if ascii\_value is Greater Than 91
120. **if** (ascii\_value < 91)
121. {
122. // Display the Word
123. cout << ptr\_array[start] << endl;
124. }
125. }
126. }
127. // Re-run the Function Increasing Start by One
128. unique\_words(ptr\_array, length, (start + 1));
129. }
130. }

Sample Output 1: unique\_word\_finder.cpp

Problem 2: array\_merger.cpp

Please Enter a String into the Program: This code was compiled in Genie Unique Words: Genie

Please Enter a String into the Program: It was on my Laptop that I wrote the code for Unique\_Word\_Finder.cpp Unique Words: Laptop Unique\_Word\_Finder.cpp

The function of this program is to take two arrays of a length given by the user, so that they may be merged into one sorted array.

Pseudocode 2: array\_merger.cpp

int main() –

1. Initialize two variables to represent the lengths of the two arrays that the user will enter.
2. Ask the user to enter the length of first array.
3. Take the value provided and have it held in the variable representing the length of the first array.
4. Initialize an array to be the first one with the length that the user specified.
5. Display a title announcing this as array one.
6. Go through each empty space in the first array.

A. Ask the user to entering in the value they want in the 000current index.

B. Take the value that the user has entered and have it 000placed at the current index of the array

1. Repeat steps 2 through 6B, but with the second array this time.
2. Initialize the length of the third array which will eventually be the merged array set it equal to the length of array one plus the length of array two.
3. Initialize an array called array three with length of value initialized in step eight.
4. Initialize a variable to represent the current location in the array and set equal to ‘0’ to begin with.
5. Go through the elements within the first array.

A. Add the current value in array one to array three at 000the location dictated by the value contained in the 000variable initialized in step ten.

B. Increment the value initialize in step ten by one.

1. Go through the elements within the second array.

A. Add the current value in array two to array three at 000the location dictated by the value contained in the 000variable initialized in step ten.

B. Increase the variable initialize in step ten by one.

1. Display the title for array three.
2. Go through the elements with array three.

A. Display the current element with a space after it.

1. Display the title to the sorted array three.
2. Run the function void bubble\_sort(int \*ptr\_array, int length); where ptr\_array is array three and length is the size of array three.

Pseudocode 2: array\_merger.cpp

Code 2: array\_merger.cpp

int main() –

1. Go through the sorted array three.

A. Display the current element in the array with a space 000at the end.

void bubble\_sort(int \*ptr\_array, int length) –

1. Initialize a variable to represent the number of passes through the list and set it equal to one.
2. Initialize a variable for length of the index and set it equal to the length.
3. For as long as the variable initialized in step one is greater than zero.

A. Set the variable initialize in step one to be zero.

B. Go through the elements within ptr\_array.

i. Check if the current value is greater than the one 000000next in the list.

- If it is do the following:

+ Increase the variable initialized in step one 0000000000by one.

+ Initialize a variable to act as a placeholder 0000000000and set it equal to the current value in the 0000000000array.

+ Set the current value in the array to be equal 0000000000to the next variable in the array.

+ Set the next value in the array to be the same 0000000000as the value held by the placeholder.

C. Decrement value initialized in step two by one.

1. //######################################################################
2. // Program Header: array\_merger.cpp
3. // The function of this program is to take two arrays of a length given
4. // by the user and combine them into one sorted array.
5. // Author: Thomas Bischoff
6. // Date: 3/31/2018
7. //######################################################################
9. #include <iostream>
11. **using** **namespace** std;
12. // Function Declaration
13. **void** bubble\_sort(**int** \*ptr\_array, **int** length);
15. **int** main()
16. {
17. // Initialize a Variable for the Length of Two Arrays
18. **int** array\_length\_one, array\_length\_two;
19. // Ask the User to Enter in the Length of the First Array
20. cout << "Please Enter in the Desired Length of Your First Array: ";
21. // Take the Value Provided and Set it to be Held by array\_length\_one
22. cin >> array\_length\_one;
23. // Intialize an Array with the Length Provided
24. **int** array\_one[array\_length\_one];
25. // Display a Title for Array One
26. cout <<  "Array One -" << endl;
27. // Add Values Provided by the User into the Array
28. **for** (**int** i = 0; i < array\_length\_one; i++)
29. {
30. // Ask the User to Enter a Value
31. cout << "Please Enter a Value for Index " << i << ": ";
32. // Take the Value Provided and Set it to be Held at the Current Index
33. cin >> array\_one[i];
34. }
35. // Ask the User to Enter in the Length of the Second Array
36. cout << "Please Enter in the Desired Length of Your Second Array: ";
37. // Take the Value Provided and Set it to be Held by array\_length\_two
38. cin >> array\_length\_two;
39. // Initialize an Array with the Length Provided
40. **int** array\_two[array\_length\_two];
41. // Display a Title for Array Two
42. cout << "Array Two -" << endl;
43. // Add Values Provided be the User into the Array
44. **for** (**int** j = 0; j < array\_length\_two; j++)
45. {
46. // Ask the User to Enter a Value
47. cout << "Please Enter a Value for Index " << j << ": ";
48. // Take the Value Provided and Set it to be Held at the Current Index
49. cin >> array\_two[j];
50. }
51. // Initialize the Length of the Third Array and Set it Equal to the Sum of the Two Other Lengths
52. **int** array\_length\_three = array\_length\_one + array\_length\_two;
53. // Initialize an Array with a Length of the Sum of the Prior Two
54. **int** array\_three[array\_length\_three];
55. // Initialize a Variable for the Position
56. **int** current\_location = 0;
57. // Go Through the First Array
58. **for** (**int** a = 0; a < array\_length\_one; a++)
59. {
60. // Add the Value from the First Array into the Third Array
61. array\_three[current\_location] = array\_one[a];
62. // Increase the Current Location by One
63. current\_location += 1;
64. }
65. // Go Through the Second Array
66. **for** (**int** b = 0; b < array\_length\_two; b++)
67. {
68. // Add the Value from the Second Array into the Thrid Array
69. array\_three[current\_location] = array\_two[b];
70. // Increase the Current Location by One
71. current\_location += 1;
72. }
73. // Display the Title for Array Three
74. cout << "Array Three -" << endl;
75. // Display the Unsorted Array
76. **for** (**int** c = 0; c < array\_length\_three; c++)
77. {
78. cout << array\_three[c] << " ";
79. }
80. cout << endl;
81. // Display the Title for the Sorted form of Array Three
82. cout << "Array Three (Sorted) -" << endl;
83. // Sort Arrray Three by Running Bubble Sort
84. bubble\_sort(array\_three, array\_length\_three);
85. // Display the Sorted Array
86. **for** (**int** d = 0; d < array\_length\_three; d++)
87. {
88. cout << array\_three[d] << " ";
89. }
90. cout << endl;
91. }
93. // Function Definition
94. **void** bubble\_sort(**int** \*ptr\_array, **int** length)
95. {
96. **int** nr\_swaps\_per\_pass = 1;
97. **int** index\_end = length;
98. **while** (nr\_swaps\_per\_pass > 0)
99. {
100. nr\_swaps\_per\_pass = 0;
101. **for** (**int** i = 0; i < index\_end; i++)
102. {
103. **if** (ptr\_array[i] > ptr\_array[i + 1])
104. {
105. nr\_swaps\_per\_pass++;
106. // Swap ptr\_array[i] with ptr\_array[i + 1]
107. **int** tmp = ptr\_array[i];
108. ptr\_array[i] = ptr\_array[i + 1];
109. ptr\_array[i + 1] = tmp;
110. }
111. }
112. index\_end --;
113. }
114. }

Sample Output 2: array\_merger.cpp

Please Enter in the Desired Length of Your First Array: 6 Array One - Please Enter a Value for Index 0: 8 Please Enter a Value for Index 1: 3 Please Enter a Value for Index 2: 12 Please Enter a Value for Index 3: 4 Please Enter a Value for Index 4: 3 Please Enter a Value for Index 5: 9 Please Enter in the Desired Length of Your Second Array: 10 Array Two - Please Enter a Value for Index 0: 4 Please Enter a Value for Index 1: 5 Please Enter a Value for Index 2: 6 Please Enter a Value for Index 3: 10 Please Enter a Value for Index 4: 13 Please Enter a Value for Index 5: 21 Please Enter a Value for Index 6: 7 Please Enter a Value for Index 7: 9 Please Enter a Value for Index 8: 4 Please Enter a Value for Index 9: 20 Array Three - 8 3 12 4 3 9 4 5 6 10 13 21 7 9 4 20 Array Three (Sorted) - 3 3 4 4 4 5 6 7 8 9 9 10 12 13 20 21