

Lab 1 Documentation

Sekhul Islam and Matthew Domenick

October 21st 2017

Table of Contents:

Class Lists.....	3
Class Text Public Member Functions.....	3
Class Text Private Attributes.....	4
Class Text Friend Functions	4
Class Text Constructors and Destructor	5
Class Text Member Functions.....	6
Test.h Source code.....	8
Lexical Source.cpp.....	9
Test.cpp.....	10
Test1.cpp.....	13
Test runs.....	18
Lexical Test.....	20

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

 Text

Public Member Functions belongs to class Text

Text (const char *charSeq="")
Text (const Text &other)
void operator= (const Text &other)
~Text ()
int getLength () const
char operator[] (int n) const
void clear ()
void showStructure () const
Text toUpper () const
Text toLower () const
bool operator== (const Text &other) const
bool operator< (const Text &other) const
bool operator> (const Text &other) const

Private Attributes:

```
int bufferSize  
char * buffer
```

BufferSize: is the private member hold size of string buffer.

Buffer: the private member variable

Friends Functions:

```
istream & operator>> (istream &input, Text &inputText)  
ostream & operator<< (ostream &output, const Text &outputText)
```

Operator << Overrides the operator "<<" so that it can output text for users

Operator >> Overrides the ">>" operator to read from file and then store in text Object.

Constructors and Destructor

◆ Text() [1/2]

```
Text::Text ( const char * charSeq = "" )
```

This constructor creates a text object containing the character sequence in the array pointed to by charSeq.

◆ Text() [2/2]

```
Text::Text ( const Text & other )
```

This function initializes the object to be an equivalent copy of other. This function is invoked automatically whenever the Text object is passed to a function using call by value, a function returns a Text object, or a Text object is initialized using another Text object.

◆ ~Text()

```
Text::~Text ( )
```

This Destructor deallocates (frees) the memory used for the implementation of the Text ADT.

Member Functions:

◆ clear()

```
void Text::clear ( )
```

This clear function clears the text objects

◆ `getLength()`

```
int Text::getLength ( ) const
```

This function returns number of characters in text object

◆ `operator<()`

```
bool Text::operator< ( const Text & other ) const
```

This function overloads operator “<” and compares two string then return true or false depend on the size.

◆ `operator=()`

```
void Text::operator= ( const Text & other )
```

This function overloads operator “=” to assign to text object

◆ `operator==()`

```
bool Text::operator== ( const Text & other ) const
```

Compares two objects

◆ operator>()

```
bool Text::operator> ( const Text & other ) const
```

Compares two object and returns true if right side is greater.

◆ operator[]()

```
char Text::operator[] ( int n ) const
```

overLoads operator “[]”

◆ showStructure()

```
void Text::showStructure ( ) const
```

This operation is used for testing/debugging, which Outputs the characters in a string.

◆ toLower()

```
Text Text::toLower ( ) const
```

This function convert lower-case of text object

◆ toUpper()

```
Text Text::toUpper ( ) const
```

This function convert lowercase to upper case of the text object

Text.h (Text Header File)

```
1
2
3 #ifndef TEXT_H
4 #define TEXT_H
5
6 #include <stdexcept>
7 #include <iostream>
8
9 using namespace std;
10
11 class Text
12 {
13 public:
14
15     // Constructors and operator=
16     Text(const char *charSeq = "");           // Initialize using char*
17     Text(const Text &other);                   // Copy constructor
18     void operator = (const Text &other);       // Assignment
19
20     // Destructor
21     ~Text();
22
23     // Text operations
24     int getLength() const;                     // # characters
25     char operator [] (int n) const;           // Subscript
26     void clear();                             // Clear string
27
28     // Output the string structure -- used in testing/debugging
29     void showStructure() const;
30
31     //-----
32     // In-lab operations
33     // toUpper/toLower operations (Programming Exercise 2)
34     Text toUpper() const;                     // Create upper-case copy
35     Text toLower() const;                     // Create lower-case copy
36
37     // Relational operations (Programming Exercise 3)
38     bool operator == (const Text& other) const;
39     bool operator < (const Text& other) const;
40     bool operator > (const Text& other) const;
41
42 private:
43
44     // Data members
45     int bufferSize; // Size of the string buffer
46     char *buffer;   // Text buffer containing a null-terminated sequence of characters
47
48     // Friends
49
50     // Text input/output operations (In-lab Exercise 1)
51     friend istream & operator >> (istream& input, Text& inputText);
52     friend ostream & operator << (ostream& output, const Text& outputText);
53
54 };
55
56 #endif
```


lexicalSoucre.cpp Main

```
8      {
9      ifstream inFile("progsamp.dat");
10     if (!inFile) {
11         cout << "Cann't open files" << endl;
12     }
13     else {
14         Text token;
15         int count = 0;
16         while (inFile >> token)
17         {
18             cout << count << " : " << token << endl;
19             count++;
20         }
21     }
22     system("pause");
23     return 0;
24 }
```

Test.cpp

```
1
2
3
4 #include "Text.h"
5 #include <iomanip>
6
7 //constructor Initializing using char
8 Text::Text(const char* charSeq)
9 {
10     bufferSize = strlen(charSeq) + 1;
11     buffer = new char[bufferSize];
12     strcpy(buffer, charSeq);
13 }
14
15 //copy constructor
16 Text::Text(const Text& other)
17 {
18     bufferSize = other.bufferSize;
19     buffer = new char[bufferSize];
20     strcpy(buffer, other.buffer);
21 }
22
23 //assignment
24 void Text::operator= (const Text &other)
25 {
26     bufferSize = other.bufferSize;
27     buffer = new char[bufferSize];
28     strcpy(buffer, other.buffer);
29 }
30
31 //Destructor
32 Text::~Text()
```

```

32 Text::~~Text()
33 {
34     delete[] buffer;
35 }
36
37 //characters operation
38 int Text::getLength() const
39 {
40     return bufferSize;
41 }
42
43 //Subscript operation
44 char Text::operator [] (int n) const
45 {
46     if (bufferSize > n)
47     {
48         return buffer[n];
49     }
50     else return '\0';
51 }
52
53 //clear string operation
54 void Text::clear()
55 {
56     delete[] buffer;
57 }
58
59 // Output the string structure -- used in testing/debugging
60 void Text::showStructure() const
61 {
62     for (int i = 0; i < bufferSize - 1; i++)
63     {
64         cout << buffer[i];
65     }
66     cout << endl;
67 }
68
69 //.....
70 //Edited on 09/21/2017
71 //Bellow function are for lab 2
72 //convert to upper case
73 Text Text::toUpper() const
74 {
75     Text temp;
76     for (int i = 0; i < bufferSize; i++)

```

```

76     for (int i = 0; i < bufferSize; i++)
77     {
78         temp.buffer[i] = toupper(buffer[i]);
79     }
80     return temp;
81 }
82
83 //convert to lower case
84 Text Text::toLowerCase() const
85 {
86     Text temp;
87     for (int i = 0; i < bufferSize; i++)
88     {
89         temp.buffer[i] = tolower(buffer[i]);
90     }
91     return temp;
92 }
93
94
95
96 // -----
97
98 istream & operator >> (istream &input, Text &inputText)
99
100 // Text input function. Extracts a string from istream input and
101 // returns it in inputText. Returns the state of the input stream.
102 {
103     const int textBufferSize = 256;    // Large (but finite)
104     char textBuffer[textBufferSize];    // text buffer
105
106                                         // Read a string into textBuffer, setw is used to prevent buffer
107                                         // overflow.
108
109     input >> setw(textBufferSize) >> textBuffer;
110
111     // Apply the Text(char*) constructor to convert textBuffer to
112     // a string. Assign the resulting string to inputText using the
113     // assignment operator.
114
115     inputText = textBuffer;
116
117     // Return the state of the input stream.
118
119     return input;
120 }
121
122
123 // -----
124
125 ostream & operator << (ostream &output, const Text &outputText)
126
127 // Text output function. Inserts outputText in ostream output.
128 // Returns the state of the output stream.
129 {
130     output << outputText.buffer;
131     return output;
132 }
133

```

Test1.cpp

```
1
2
3 //-----
4 //
5 //  Laboratory 1                                test1.cpp
6 //
7 //  Test program for the operations in the Text ADT
8 //
9 //-----
10
11 #include <iostream>
12 #include "Text.h"
13
14 //-----
15 //
16 //  Function prototype
17
18 void copyTester(Text copyText); // copyText is passed by value
19 void print_help();
20
21 //-----
22
23 int main()
24 {
25     Text a("a"), // Predefined test text objects
26         alp("alp"),
27         alpha("alpha"),
28         epsilon("epsilon"),
29         empty,
30         assignText, // Destination for assignment
31         inputText; // Input text object
32     int n; // Input subscript
33     char ch; // Character specified by subscript
34     selection; // Input test selection
35
36     // Get user test selection.
37     print_help();
38
39     // Execute the selected test.
40     cin >> selection;
41
42     cout << endl;
43     switch (selection)
```

```

43     switch (selection)
44     {
45     case '1':
46         // Test 1 : Tests the constructors.
47         cout << "Structure of various text objects: " << endl;
48         cout << "text object: alpha" << endl;
49         alpha.showStructure();
50         cout << "text object: epsilon" << endl;
51         epsilon.showStructure();
52         cout << "text object: a" << endl;
53         a.showStructure();
54         cout << "empty text object" << endl;
55         empty.showStructure();
56         break;
57     case '2':
58         // Test 2 : Tests the length operation.
59         cout << "Lengths of various text object:" << endl;
60         cout << " alpha   : " << alpha.getLength() << endl;
61         cout << " epsilon : " << epsilon.getLength() << endl;
62         cout << " a       : " << a.getLength() << endl;
63         cout << " empty   : " << empty.getLength() << endl;
64         break;
65
66     case '3':
67         // Test 3 : Tests the subscript operation.
68         cout << "Enter a subscript : ";
69         cin >> n;
70         ch = alpha[n];
71         cout << " alpha[" << n << "] : ";
72         if (ch == '\0')
73             cout << "\\0" << endl;
74         else
75             cout << ch << endl;
76         break;
77
78     case '4':
79         // Test 4 : Tests the assignment and clear operations.
80         cout << "Assignments:" << endl;
81         cout << "assignText = alpha" << endl;
82         assignText = alpha;
83         assignText.showStructure();
84         cout << "assignText = a" << endl;
85         assignText = a;
86         assignText.showStructure();
87         cout << "assignText = empty" << endl;
88         assignText = empty;

```

```

88     assignText = empty;
89     assignText.showStructure();
90     cout << "assignText = epsilon" << endl;
91     assignText = epsilon;
92     assignText.showStructure();
93     cout << "assignText = assignText" << endl;
94     assignText = assignText;
95     assignText.showStructure();
96     cout << "assignText = alpha" << endl;
97     assignText = alpha;
98     assignText.showStructure();
99     cout << "Clear assignText" << endl;
100    assignText.clear();
101    assignText.showStructure();
102    cout << "Confirm that alpha has not been cleared" << endl;
103    alpha.showStructure();
104    break;
105
106    case '5':
107        // Test 5 : Tests the copy constructor and operator= operations.
108        cout << "Calls by value:" << endl;
109        cout << "alpha before call" << endl;
110        alpha.showStructure();
111        copyTester(alpha);
112        cout << "alpha after call" << endl;
113        alpha.showStructure();
114
115        cout << "a before call" << endl;
116        a.showStructure();
117        a = epsilon;
118        cout << "a after call" << endl;
119        a.showStructure();
120        cout << "epsilon after call" << endl;
121        epsilon.showStructure();
122        break;
123
124    #if LAB1_TEST1
125        case '6': // In-lab Exercise 2
126                // Test 6 : Tests toUpper and toLower
127                cout << "Testing toUpper and toLower."
128                    << "Enter a mixed case string: " << endl;
129                cin >> inputText;
130                cout << "Input string:" << endl;
131                inputText.showStructure();
132                cout << "Upper case copy: " << endl;
133                inputText.toUpper().showStructure();

```

```

133     inputText.toUpper().showStructure();
134     cout << "Lower case copy: " << endl;
135     inputText.toLower().showStructure();
136     break;
137 #endif // LAB1_TEST1
138
139 #if LAB1_TEST2
140     case '7': // In-lab Exercise 3
141               // Test 7 : Tests the relational operations.
142     cout << " left      right      <  ==  > " << endl;
143     cout << "-----" << endl;
144     cout << " alpha    epsilon    " << (alpha<epsilon)
145           << " " << (alpha == epsilon) << " "
146           << (alpha>epsilon) << endl;
147     cout << " epsilon  alpha      " << (epsilon<alpha)
148           << " " << (epsilon == alpha) << " "
149           << (epsilon>alpha) << endl;
150     cout << " alpha    alpha      " << (alpha<alpha) << " "
151           << (alpha == alpha) << " " << (alpha>alpha) << endl;
152     cout << " alp     alpha      " << (alp<alpha) << " "
153           << (alp == alpha) << " " << (alp>alpha) << endl;
154     cout << " alpha    alp        " << (alpha<alp) << " "
155           << (alpha == alp) << " " << (alpha>alp) << endl;
156     cout << " a        alpha      " << (a<alpha) << " "
157           << (a == alpha) << " " << (a>alpha) << endl;
158     cout << " alpha    a          " << (alpha<a) << " "
159           << (alpha == a) << " " << (alpha>a) << endl;
160     cout << " empty   alpha      " << (empty<alpha) << " "
161           << (empty == alpha) << " " << (empty>alpha) << endl;
162     cout << " alpha    empty      " << (alpha<empty) << " "
163           << (alpha == empty) << " " << (alpha>empty) << endl;
164     cout << " empty   empty      " << (empty<empty) << " "
165           << (empty == empty) << " " << (empty>empty) << endl;
166     break;
167 #endif // LAB1_TEST2
168
169     default:
170         cout << "" << selection << " specifies an inactive or invalid test" << endl;
171     }
172     system("pause");
173     return 0;
174 }
175
176 //-----

```



```

176 //-----
177
178 void copyTester(Text copyText)
179
180 // Dummy routine that is passed a text object using call by value. Outputs
181 // copyText and clears it.
182
183 {
184     cout << "Copy of text object" << endl;
185     copyText.showStructure();
186     cout << "Clear copy" << endl;
187     copyText.clear();
188     copyText.showStructure();
189 }
190
191 //-----
192
193 void print_help()
194 {
195     cout << endl << "Tests:" << endl;
196     cout << "  1  Tests the constructors" << endl;
197     cout << "  2  Tests the length operation" << endl;
198     cout << "  3  Tests the subscript operation" << endl;
199     cout << "  4  Tests the assignment and clear operations" << endl;
200     cout << "  5  Tests the copy constructor and operator= operations" << endl;
201
202     cout << "  6  Tests the toUpper and toLower operations      "
203 #if LAB1_TEST1
204     << "(Active   : "
205 #else
206     << "(Inactive : "
207 #endif // LAB1_TEST1
208     << "In-lab Exercise 2)" << endl;
209
210     cout << "  7  Tests the relational operations      "
211 #if LAB1_TEST2
212     << "              (Active   : "
213 #else
214     << "              (Inactive : "
215 #endif // LAB1_TEST2
216     << "In-lab Exercise 3)" << endl;
217     cout << "Select the test to run : ";
218 }

```

Test runs

Testing menu

```
Tests:
1 Tests the constructors
2 Tests the length operation
3 Tests the subscript operation
4 Tests the assignment and clear operations
5 Tests the copy constructor and operator= operations
6 Tests the toUpper and toLower operations (Inactive : In-lab Exercise 2)
7 Tests the relational operations (Inactive : In-lab Exercise 3)
Select the test to run : 1
```

Testing the constructors

```
C:\Users\matthew\Documents\Visual Studio 2015\Projects\Project24\Debug\Project24.exe

Tests:
1 Tests the constructors
2 Tests the length operation
3 Tests the subscript operation
4 Tests the assignment and clear operations
5 Tests the copy constructor and operator= operations
6 Tests the toUpper and toLower operations (Inactive : In-lab Exercise 2)
7 Tests the relational operations (Inactive : In-lab Exercise 3)
Select the test to run : 1

Structure of various text objects:
text object: alpha
alpha
text object: epsilon
epsilon
text object: a
a
empty text object

Press any key to continue . . .
```

Testing the length operation

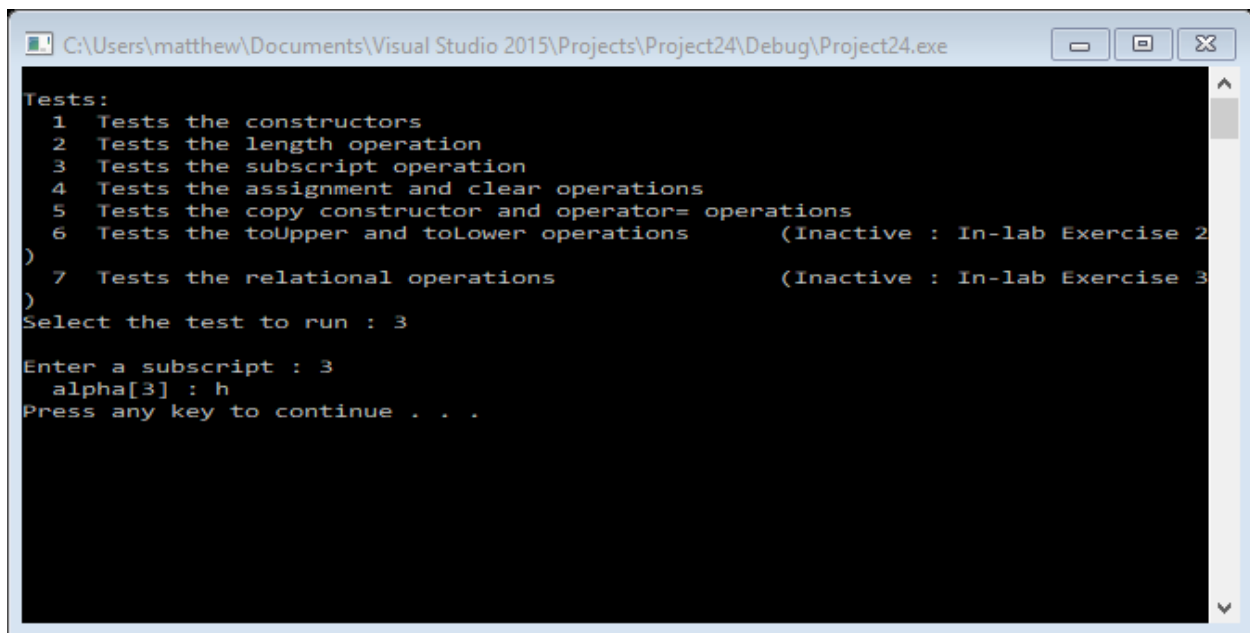
```
C:\Users\matthew\Documents\Visual Studio 2015\Projects\Project24\Debug\Project24.exe

Tests:
1 Tests the constructors
2 Tests the length operation
3 Tests the subscript operation
4 Tests the assignment and clear operations
5 Tests the copy constructor and operator= operations
6 Tests the toUpper and toLower operations (Inactive : In-lab Exercise 2)
7 Tests the relational operations (Inactive : In-lab Exercise 3)
Select the test to run : 2

Lengths of various text object:
alpha : 6
epsilon : 8
a : 2
empty : 1

Press any key to continue . . .
```

Testing the subscript operation

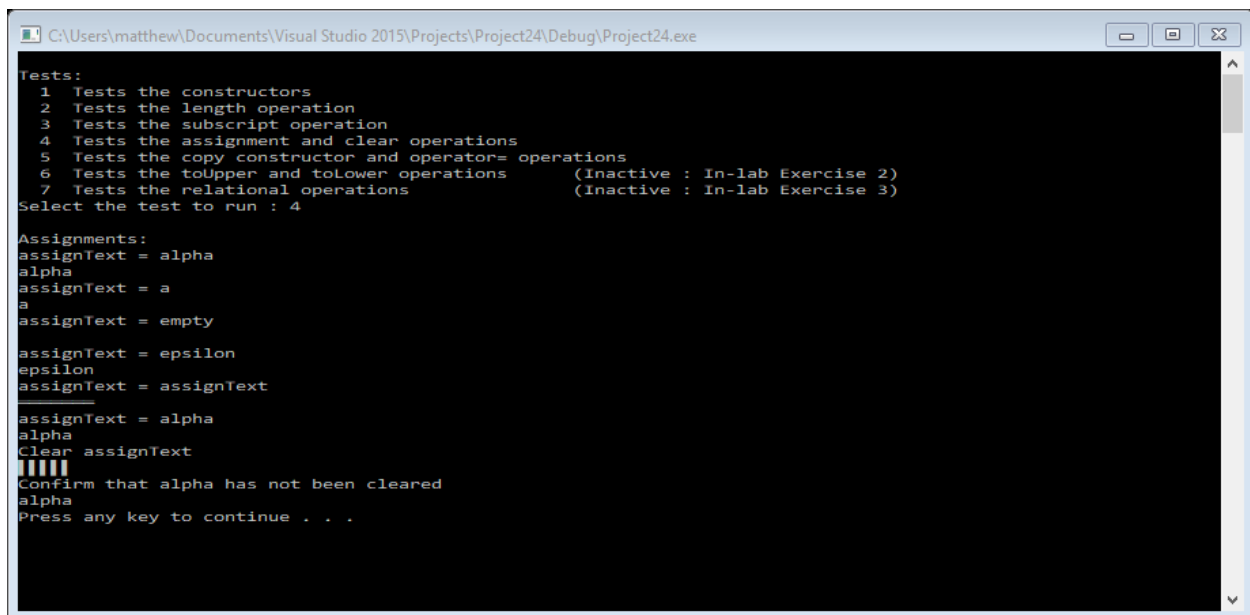


```
C:\Users\matthew\Documents\Visual Studio 2015\Projects\Project24\Debug\Project24.exe

Tests:
1 Tests the constructors
2 Tests the length operation
3 Tests the subscript operation
4 Tests the assignment and clear operations
5 Tests the copy constructor and operator= operations
6 Tests the toUpper and toLower operations (Inactive : In-lab Exercise 2)
7 Tests the relational operations (Inactive : In-lab Exercise 3)
)
)
Select the test to run : 3

Enter a subscript : 3
alpha[3] : h
Press any key to continue . . .
```

Testing the assignment and clear operations

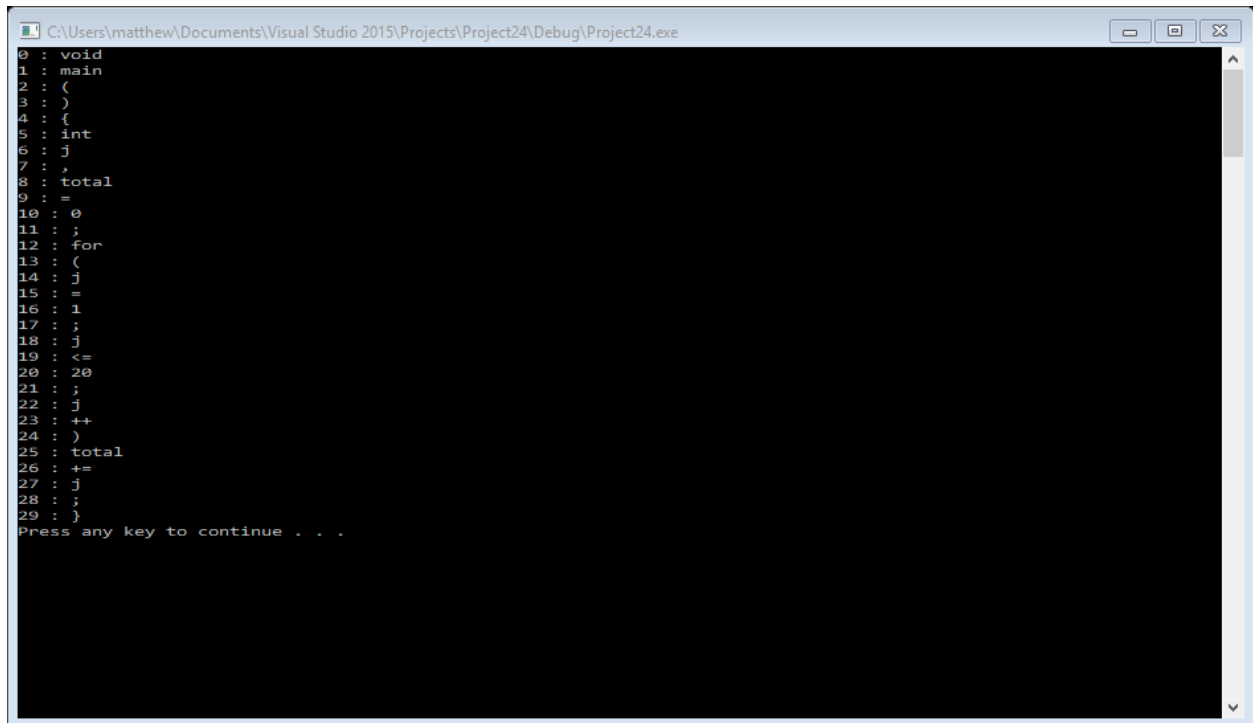


```
C:\Users\matthew\Documents\Visual Studio 2015\Projects\Project24\Debug\Project24.exe

Tests:
1 Tests the constructors
2 Tests the length operation
3 Tests the subscript operation
4 Tests the assignment and clear operations
5 Tests the copy constructor and operator= operations
6 Tests the toUpper and toLower operations (Inactive : In-lab Exercise 2)
7 Tests the relational operations (Inactive : In-lab Exercise 3)
)
)
Select the test to run : 4

Assignments:
assignText = alpha
alpha
assignText = a
a
assignText = empty
assignText = epsilon
epsilon
assignText = assignText
assignText = alpha
alpha
Clear assignText
|||||
Confirm that alpha has not been cleared
alpha
Press any key to continue . . .
```

Lexcial Test



A screenshot of a Visual Studio 2015 debug console window. The title bar shows the file path: C:\Users\matthew\Documents\Visual Studio 2015\Projects\Project24\Debug\Project24.exe. The console displays the following C++ code with line numbers 0 through 29:

```
0 : void
1 : main
2 : (
3 : )
4 : {
5 : int
6 : j
7 : ,
8 : total
9 : =
10 : 0
11 : ;
12 : for
13 : (
14 : j
15 : =
16 : 1
17 : ;
18 : j
19 : <=
20 : 20
21 : ;
22 : j
23 : ++
24 : )
25 : total
26 : +=
27 : j
28 : ;
29 : }
```

Below the code, the text "Press any key to continue . . ." is displayed. The console window has a scrollbar on the right side.