Lab 1 Documentation

Sekhul Islam and Matthew Domenick

October 21st 2017

Table of Contents:

Class List

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

C 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 veriable

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)

```cpp


#ifndef TEXT_H
#define TEXT_H

#include <stdexcept>
#include <iostream>

using namespace std;

class Text
{
public:

    // Constructors and operator=
    Text(const char *charSeq = "");                // Initialize using char*
    Text(const Text &other);                       // Copy constructor
    void operator = (const Text &other);            // Assignment

    // Destructor
    ~Text();

    // Text operations
    int getLength() const;                           // # characters
    char operator [] (int n) const;                 // Subscript
    void clear();                                    // Clear string

    // Output the string structure -- used in testing/debugging
    void showStructure() const;

    //-----------------------------------------------------------------
    // In-lab operations
    // toUpper/toLower operations (Programming Exercise 2)
    Text toUpper() const;                            // Create upper-case copy
    Text toLower() const;                            // Create lower-case copy

    // Relational operations (Programming Exercise 3)
    bool operator == (const Text& other) const;
    bool operator <  (const Text& other) const;
    bool operator >  (const Text& other) const;

private:

    // Data members
    int bufferSize;   // Size of the string buffer
    char *buffer;     // Text buffer containing a null-terminated sequence of characters

    // Friends

    // Text input/output operations (In-lab Exercise 1)
    friend istream & operator >> (istream& input, Text& inputText);
    friend ostream & operator << (ostream& output, const Text& outputText);

};

#endif
```

lexicalSoucre.cpp Main

```
            {
    ifstream inFile("progsamp.dat");
    if (!inFile) {
        cout << "Cann't open files" << endl;
    }
    else {
        Text token;
        int count = 0;
        while (inFile >> token)
        {
            cout << count << " : " << token << endl;
            count++;
        }
    }
    system("pause");
    return 0;
}
```

Test.cpp

```cpp
#include "Text.h"
#include <iomanip>

//constructor Initializing using char
Text::Text(const char* charSeq)
{
    bufferSize = strlen(charSeq) + 1;
    buffer = new char[bufferSize];
    strcpy(buffer, charSeq);
}

//copy constructor
Text::Text(const Text& other)
{
    bufferSize = other.bufferSize;
    buffer = new char[bufferSize];
    strcpy(buffer, other.buffer);
}

//assignment
void Text::operator= (const Text &other)
{
    bufferSize = other.bufferSize;
    buffer = new char[bufferSize];
    strcpy(buffer, other.buffer);
}

//Destructor
Text::~Text()
```

```
Text::~Text()
{
    delete[] buffer;
}

//#characters operation
int Text::getLength() const
{
    return bufferSize;
}

//Subcript operation
char Text::operator [] (int n) const
{
    if (bufferSize > n)
    {
        return buffer[n];
    }
    else return '\0';
}

//clear string operation
void Text::clear()
{
    delete[] buffer;
}

// Output the string structure -- used in testing/debugging
void Text::showStructure() const
{
    for (int i = 0; i < bufferSize - 1; i++)
    {
        cout << buffer[i];
    }
    cout << endl;
}

//.............................................................
//Edited on 09/21/2017
//Bellow function are for lab 2
//convert to upper case
Text Text::toUpper() const
{
    Text temp;
    for (int i = 0; i < bufferSize; i++)
```

```
    for (int i = 0; i < bufferSize; i++)
    {
        temp.buffer[i] = toupper(buffer[i]);
    }
    return temp;
}

//convert to lower case
Text Text::toLower() const
{
    Text temp;
    for (int i = 0; i < bufferSize; i++)
    {
        temp.buffer[i] = tolower(buffer[i]);
    }
    return temp;
}



// ----------------------------------------------------------------

istream & operator >> (istream &input, Text &inputText)

// Text input function. Extracts a string from istream input and
// returns it in inputText. Returns the state of the input stream.

{
    const int textBufferSize = 256;     // Large (but finite)
    char textBuffer[textBufferSize];   // text buffer

                                       // Read a string into textBuffer, setw is used to prevent buffer
                                       // overflow.

    input >> setw(textBufferSize) >> textBuffer;

    // Apply the Text(char*) constructor to convert textBuffer to
    // a string. Assign the resulting string to inputText using the
    // assignment operator.

    inputText = textBuffer;

    // Return the state of the input stream.

    return input;
}
```

```
}

//-----------------------------------------------------------------------

ostream & operator << (ostream &output, const Text &outputText)

// Text output function. Inserts outputText in ostream output.
// Returns the state of the output stream.

{
    output << outputText.buffer;
    return output;
}
```

Test1.cpp

```cpp


//-----------------------------------------------------------------
//
//  Laboratory 1                                          test1.cpp
//
//  Test program for the operations in the Text ADT
//
//-----------------------------------------------------------------

#include <iostream>
#include "Text.h"

//-----------------------------------------------------------------
//
//  Function prototype

void copyTester(Text copyText);   // copyText is passed by value
void print_help();

//-----------------------------------------------------------------

int main()
{
    Text a("a"),                // Predefined test text objects
        alp("alp"),
        alpha("alpha"),
        epsilon("epsilon"),
        empty,
        assignText,          // Destination for assignment
        inputText;           // Input text object
    int n;                        // Input subscript
    char ch,                      // Character specified by subscript
        selection;               // Input test selection

                                 // Get user test selection.
    print_help();

    // Execute the selected test.
    cin >> selection;

    cout << endl;
    switch (selection)
```

```
    switch (selection)
    {
    case '1':
        // Test 1 : Tests the constructors.
        cout << "Structure of various text objects: " << endl;
        cout << "text object: alpha" << endl;
        alpha.showStructure();
        cout << "text object: epsilon" << endl;
        epsilon.showStructure();
        cout << "text object: a" << endl;
        a.showStructure();
        cout << "empty text object" << endl;
        empty.showStructure();
        break;
    case '2':
        // Test 2 : Tests the length operation.
        cout << "Lengths of various text object:" << endl;
        cout << " alpha   : " << alpha.getLength() << endl;
        cout << " epsilon : " << epsilon.getLength() << endl;
        cout << " a       : " << a.getLength() << endl;
        cout << " empty   : " << empty.getLength() << endl;
        break;

    case '3':
        // Test 3 : Tests the subscript operation.
        cout << "Enter a subscript : ";
        cin >> n;
        ch = alpha[n];
        cout << "  alpha[" << n << "] : ";
        if (ch == '\0')
            cout << "\\0" << endl;
        else
            cout << ch << endl;
        break;

    case '4':
        // Test 4 : Tests the assignment and clear operations.
        cout << "Assignments:" << endl;
        cout << "assignText = alpha" << endl;
        assignText = alpha;
        assignText.showStructure();
        cout << "assignText = a" << endl;
        assignText = a;
        assignText.showStructure();
        cout << "assignText = empty" << endl;
        assignText = empty;
```

```
        assignText = empty;
        assignText.showStructure();
        cout << "assignText = epsilon" << endl;
        assignText = epsilon;
        assignText.showStructure();
        cout << "assignText = assignText" << endl;
        assignText = assignText;
        assignText.showStructure();
        cout << "assignText = alpha" << endl;
        assignText = alpha;
        assignText.showStructure();
        cout << "Clear assignText" << endl;
        assignText.clear();
        assignText.showStructure();
        cout << "Confirm that alpha has not been cleared" << endl;
        alpha.showStructure();
        break;

    case '5':
        // Test 5 : Tests the copy constructor and operator= operations.
        cout << "Calls by value:" << endl;
        cout << "alpha before call" << endl;
        alpha.showStructure();
        copyTester(alpha);
        cout << "alpha after call" << endl;
        alpha.showStructure();

        cout << "a before call" << endl;
        a.showStructure();
        a = epsilon;
        cout << "a after call" << endl;
        a.showStructure();
        cout << "epsilon after call" << endl;
        epsilon.showStructure();
        break;

#if   LAB1_TEST1
    case '6':                                   // In-lab Exercise 2
                                                // Test 6 : Tests toUpper and toLower
        cout << "Testing toUpper and toLower."
            << "Enter a mixed case string: " << endl;
        cin >> inputText;
        cout << "Input string:" << endl;
        inputText.showStructure();
        cout << "Upper case copy: " << endl;
        inputText.toUpper().showStructure();
```

```
        inputText.toUpper().showStructure();
        cout << "Lower case copy: " << endl;
        inputText.toLower().showStructure();
        break;
#endif // LAB1_TEST1

#if   LAB1_TEST2
    case '7':                                   // In-lab Exercise 3
                                                // Test 7 : Tests the relational operations.
        cout << "  left     right     <   ==   > " << endl;
        cout << "--------------------------------" << endl;
        cout << " alpha    epsilon    " << (alpha<epsilon)
            << "    " << (alpha == epsilon) << "   "
            << (alpha>epsilon) << endl;
        cout << " epsilon   alpha     " << (epsilon<alpha)
            << "    " << (epsilon == alpha) << "   "
            << (epsilon>alpha) << endl;
        cout << " alpha     alpha     " << (alpha<alpha) << "    "
            << (alpha == alpha) << "   " << (alpha>alpha) << endl;
        cout << "  alp      alpha     " << (alp<alpha) << "    "
            << (alp == alpha) << "   " << (alp>alpha) << endl;
        cout << " alpha      alp      " << (alpha<alp) << "    "
            << (alpha == alp) << "   " << (alpha>alp) << endl;
        cout << "   a       alpha     " << (a<alpha) << "    "
            << (a == alpha) << "   " << (a>alpha) << endl;
        cout << " alpha       a       " << (alpha<a) << "    "
            << (alpha == a) << "   " << (alpha>a) << endl;
        cout << " empty     alpha     " << (empty<alpha) << "    "
            << (empty == alpha) << "   " << (empty>alpha) << endl;
        cout << " alpha     empty     " << (alpha<empty) << "    "
            << (alpha == empty) << "   " << (alpha>empty) << endl;
        cout << " empty     empty     " << (empty<empty) << "    "
            << (empty == empty) << "   " << (empty>empty) << endl;
        break;
#endif // LAB1_TEST2

    default:
        cout << "'" << selection << "' specifies an inactive or invalid test" << endl;
    }
    system("pause");
    return 0;
}

//-----------------------------------------------------------------
```

```
//-----------------------------------------------------------------

void copyTester(Text copyText)

// Dummy routine that is passed a text object using call by value. Outputs
// copyText and clears it.

{
    cout << "Copy of text object" << endl;
    copyText.showStructure();
    cout << "Clear copy" << endl;
    copyText.clear();
    copyText.showStructure();
}

//-----------------------------------------------------------------

void print_help()
{
    cout << endl << "Tests:" << endl;
    cout << "  1  Tests the constructors" << endl;
    cout << "  2  Tests the length operation" << endl;
    cout << "  3  Tests the subscript operation" << endl;
    cout << "  4  Tests the assignment and clear operations" << endl;
    cout << "  5  Tests the copy constructor and operator= operations" << endl;

    cout << "  6  Tests the toUpper and toLower operations      "
#if LAB1_TEST1
        << "(Active   : "
#else
        << "(Inactive : "
#endif  // LAB1_TEST1
        << "In-lab Exercise 2)" << endl;

    cout << "  7  Tests the relational operations    "
#if LAB1_TEST2
        << "           (Active   : "
#else
        << "           (Inactive : "
#endif  // LAB1_TEST2
        << "In-lab Exercise 3)" << endl;
    cout << "Select the test to run : ";
}
```

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