Phong Nguyen

Razi Khanom

CSC 2010

Laboratory 1.0

**Contents**

#### [Class Index](#_bookmark0) 5

* 1. [Class List](#_bookmark1) 5

#### [File Index](#_bookmark2) 7

* 1. [File List](#_bookmark3) 7

#### [Class Documentation](#_bookmark4) 9

* 1. [Text Class Reference](#_bookmark6) 5
     1. [Constructor & Destructor Documentation](#_bookmark8) 6

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

|  |  |  |
| --- | --- | --- |
|  | [3.1.1.1](#_bookmark9) | [Text() [1/2]](#_bookmark9) . . . . . |
| [3.1.1.2](#_bookmark12) | [Text() [2/2]](#_bookmark12) . . . . . |
| [3.1.2](#_bookmark16) | [3.1.1.3](#_bookmark13)  [Member](#_bookmark16) | [∼Text()](#_bookmark13) . . . . . . . .  [Function Documentation](#_bookmark16) |

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7 [3.1.2.1 clear()](#_bookmark17) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

* + - 1. [getLength()](#_bookmark19) 7
      2. [operator*<*()](#_bookmark23) 8
      3. [operator=()](#_bookmark24) 8
      4. [operator==()](#_bookmark26) 8
      5. [operator*>*()](#_bookmark28) 8
      6. [operator[]()](#_bookmark29) 8
      7. [showStructure()](#_bookmark33) 9
      8. [toLower()](#_bookmark35) 9
      9. [toUpper()](#_bookmark37) 9

. . . . . . . . . . . . . . . . . . . . . . . . 9

|  |  |  |
| --- | --- | --- |
| [3.1.3](#_bookmark39) | [Friends](#_bookmark39) | [And Related Function Documentation](#_bookmark39) |
|  | [3.1.3.1](#_bookmark42) | [operator*<<*](#_bookmark42). . . . . . . . . . . . |
|  | [3.1.3.2](#_bookmark43) | [operator*>>*](#_bookmark43). . . . . . . . . . . . |
| [3.1.4](#_bookmark45) | [Member](#_bookmark45) | [Data Documentation](#_bookmark45) . . . . . . . . . |
|  | [3.1.4.1](#_bookmark46) | [buffer](#_bookmark46) . . . . . . . . . . . . . . . . |
|  | [3.1.4.2](#_bookmark48) | [bufferSize](#_bookmark48) . . . . . . . . . . . . . |

. . . . . . . . . . . . . . . . . . . . . . . . 10

. . . . . . . . . . . . . . . . . . . . . . . . 10

. . . . . . . . . . . . . . . . . . . . . . . . 10

. . . . . . . . . . . . . . . . . . . . . . . . 10

. . . . . . . . . . . . . . . . . . . . . . . . 10

**ii CONTENTS**

1. [File Documentation](#_bookmark50) 11
   1. [E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/config.h File Reference](#_bookmark52) 11
      1. [Macro Definition Documentation](#_bookmark53) 11
         1. [LAB1\_TEST1](#_bookmark54) 11
         2. [LAB1\_TEST2](#_bookmark58) 12
   2. [E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/lexical.cpp File Reference](#_bookmark60) 12
      1. [Function Documentation](#_bookmark61) 12

[4.2.1.1 main()](#_bookmark62) 12

* 1. [E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/test1.cpp File Reference](#_bookmark66) 13
     1. [Function Documentation](#_bookmark67) 13
        1. [copyTester()](#_bookmark68) 13

[4.3.1.2 main()](#_bookmark71) 14

[4.3.1.3 print\_help()](#_bookmark74) 15

* 1. [E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/text.cpp File Reference](#_bookmark76) 15
     1. [Function Documentation](#_bookmark79) 16
        1. [operator*<<*()](#_bookmark80) 16
        2. [operator*>>*()](#_bookmark82) 16
  2. [E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/Text.h File Reference](#_bookmark86) 17
  3. [E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/textio.cpp File Reference](#_bookmark87) 17
     1. [Function Documentation](#_bookmark90) 18
        1. [operator*<<*()](#_bookmark91) 18
        2. [operator*>>*()](#_bookmark93) 18

[Index](#_bookmark95) 19

1. [Attachments](#cover_sheet) 24
   1. Lab [Report 24](#cover_sheet)
      1. [Cover Sheet](#cover_sheet) 24

**iii CONTENTS**

* + 1. [Implementation Testing](#Implementation_Testing) 25
    2. [Programming Exercise 1](#Programming_excerse) 28
    3. [Analysis Exercise 1](#Analysis_Exercise) 29
  1. [Program Output](#Test_plan1)  [30](#cover_sheet)
     1. [Test Plan 1-1 (constructors)](#Test_plan1) 30
     2. [Test Plan 1-2 (length operation)](#Test_plan12) 30
     3. [Test Plan 1-3 (subscript operation)](#Test_plan13) 31
     4. [Test Plan 1-4 (assignment and clear operations)](#Test_plan14) 31
     5. [Test Plan 1-5 (lexical analysis program)](#Test_plan15) 32
  2. [Source Codes](#text_h)  [33](#cover_sheet)
     1. [Text.h](#text_h) 33
     2. [Text1.cpp](#text1) 34
     3. [Text.cpp](#text) 38
     4. [Lexical.cpp](#lexical) 41

**Chapter 1**

**Class Index**

**1.1 Class List**

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

[Text](#_bookmark6) [5](#_bookmark6)

**Class Index**

**Chapter 2**

**File Index**

**2.1 File List**

Here is a list of all files with brief descriptions:

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[config.h](#_bookmark52) [11](#_bookmark52)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[lexical.cpp](#_bookmark59) [12](#_bookmark60)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[test1.cpp](#_bookmark64) [13](#_bookmark66)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[text.cpp](#_bookmark77) [15](#_bookmark76)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[Text.h](#_bookmark84) [17](#_bookmark86)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[textio.cpp](#_bookmark88) [17](#_bookmark87)

**4 File Index**

**Chapter 3**

**Class Documentation**

* 1. **Text Class Reference**

#include *<*Text.h*>*

**Public Member Functions**

* + - [Text](#_bookmark10) (const char ∗charSeq="")

*Constructors and operator=.*

* + - [Text](#_bookmark11) (const [Text](#_bookmark6) &other)

*Initialize using char*∗*.*

* + - void [operator=](#_bookmark25) (const [Text](#_bookmark6) &other)

*Copy constructor.*

* + - [∼Text](#_bookmark14) ()

*Assignment.*

* + - int [getLength](#_bookmark20) () const

[*Text*](#_bookmark6) *operations.*

* + - char [operator[ ]](#_bookmark30) (int n) const

**characters**

* + - void [clear](#_bookmark18) ()

*Subscript.*

* + - void [showStructure](#_bookmark34) () const

*Clear string.*

* + - [Text](#_bookmark6) [toUpper](#_bookmark38) () const
    - [Text](#_bookmark6) [toLower](#_bookmark36) () const

*Create upper-case copy.*

* + - bool [operator==](#_bookmark27) (const [Text](#_bookmark6) &other) const

*Create lower-case copy.*

* + - bool [operator*<*](#_bookmark21)(const [Text](#_bookmark6) &other) const
    - bool [operator*>*](#_bookmark31)(const [Text](#_bookmark6) &other) const

**6 Class Documentation**

**Private Attributes**

* int [bufferSize](#_bookmark49)

*Data members.*

* char ∗ [buffer](#_bookmark47)

*Size of the string buffer.*

**Friends**

* istream & [operator*>>*](#_bookmark44)(istream &input, [Text](#_bookmark6) &inputText)

[*Text*](#_bookmark6) *buffer containing a null-terminated sequence of characters.*

* ostream & [operator*<<*](#_bookmark40)(ostream &output, const [Text](#_bookmark6) &outputText)

**3.1.1 Constructor & Destructor Documentation**

**3.1.1.1 Text()** [1/2]

Text::Text (

const char ∗ *charSeq = ""* )

Constructors and operator=.

Creates a string containing the delimited sequence of characters charSeq. Allocates enough memory for this string. Copy the string

**3.1.1.2 Text()** [2/2]

Text::Text (

const [Text](#_bookmark6) & *other* )

Initialize using char∗. Copy constructor.

**3.1.1.3** ∼**Text()** Text::∼Text ( ) Assignment.

Frees the memory used by the [Text](#_bookmark6) object buffer.

Destructor

* 1. **Text Class Reference** **7**
     1. **Member Function Documentation**
        1. **clear()**

void Text::clear ( )

Subscript.

Clears a [Text](#_bookmark6) object – i.e., makes it empty. The buffer size remains unchanged. Here is the caller graph for this function:



main

Text::clear

copyTester

* + - 1. **getLength()**

int Text::getLength ( ) const

[Text](#_bookmark6) operations.

Returns the number of characters in the [Text](#_bookmark6) object buffer (excluding the null character). Here is the caller graph for this function:



main

Text::getLength

Text::operator=

**8 Class Documentation**

* + - 1. **operator***<***()**

bool Text::operator*<* (

const [Text](#_bookmark6) & *other* ) const

* + - 1. **operator=()**

void Text::operator= (

const [Text](#_bookmark6) & *other* )

Copy constructor.

Assigns other to a [Text](#_bookmark6) object. Here is the call graph for this function:



Text::operator=

Text::getLength

* + - 1. **operator==()**

bool Text::operator== (

const [Text](#_bookmark6) & *other* ) const

Create lower-case copy.

Relational operations (Programming Exercise 3)

* + - 1. **operator***>***()**

bool Text::operator*>* (

const [Text](#_bookmark6) & *other* ) const

* + - 1. **operator[]()**

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

**characters**

Returns the nth character in a [Text](#_bookmark6) object – where the characters are numbered beginning with zero.

If the subscript is out of range, return an empty character (/0)

**3.1 Text Class Reference** **9**

* + - 1. **showStructure()**

void Text::showStructure ( ) const

Clear string.

Output the string structure – used in testing/debugging

Outputs the characters in a string. This operation is intended for testing/debugging purposes only. Here is the caller graph for this function:



main

Text::showStructure

copyTester

* + - 1. **toLower()**

[Text](#_bookmark6) Text::toLower ( ) const

Create upper-case copy.

* + - 1. **toUpper()**

[Text](#_bookmark6) Text::toUpper ( ) const

In-lab operations toUpper/toLower operations (Programming Exercise 2)

**3.1.3 Friends And Related Function Documentation**

**10 Class Documentation**

* + - 1. **operator***<<*

ostream& operator*<<* (

ostream & *output,*

const [Text](#_bookmark6) & *outputText* ) [friend]

[Text](#_bookmark6) output function. Inserts outputText in ostream output. Returns the state of the output stream.

* + - 1. **operator***>>*

istream& operator*>>* (

istream & *input,*

[Text](#_bookmark6) & *inputText* ) [friend]

[Text](#_bookmark6) buffer containing a null-terminated sequence of characters. Friends [Text](#_bookmark6) input/output operations (In-lab Exercise 1)

[Text](#_bookmark6) input function. Extracts a string from istream input and returns it in inputText. Returns the state of the input stream. Large (but finite)

text buffer

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

Apply the [Text(char )](#_bookmark6) constructor to convert textBuffer to a string. Assign the resulting string to inputText us- ing the assignment operator.

[∗](#_bookmark6)

Return the state of the input stream.

* + 1. **Member Data Documentation**
       1. **buffer**

char∗ Text::buffer [private]

Size of the string buffer.

* + - 1. **bufferSize**

int Text::bufferSize [private]

Data members.

The documentation for this class was generated from the following files:

* + - * + E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[Text.h](#_bookmark84)
        + E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/[text.cpp](#_bookmark77)

**Chapter 4**

**File Documentation**

## E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/config.h File Reference

This graph shows which files directly or indirectly include this file:



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/test1.cpp

E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/config.h

**Macros**

* + - #define [LAB1\_TEST1](#_bookmark55) 0
    - #define [LAB1\_TEST2](#_bookmark56) 0

### Macro Definition Documentation

#### LAB1\_TEST1

#define LAB1\_TEST1 0

[Text](#_bookmark6) class (Lab 1) configuration file. Activate test 'N' by defining the corresponding LAB1\_TESTN to have the value 1.

**12 File Documentation**

#### LAB1\_TEST2

#define LAB1\_TEST2 0

## E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/lexical.cpp File Reference

#include *<*fstream*>*

#include *<*iostream*>*

#include "Text.h"

Include dependency graph for lexical.cpp:



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/lexical.cpp

fstream

Text.h

iostream

stdexcept

**Functions**

* int [main](#_bookmark63) ()

### Function Documentation

#### main()

int main ( )

Project Name : Lab 1-Exercise 1

Developer Name : Phong Nguyen

Date : 01/19/2018

Description : Lexical Analysis Program Token

Open the specified program file. Counts tokens

**4.3 E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/test1.cpp File Reference** **13**

## E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/test1.cpp File Reference

#include *<*iostream*>*

#include "Text.h"

#include "config.h"

Include dependency graph for test1.cpp:



Text.h

config.h

iostream

stdexcept

|  |  |
| --- | --- |
| E:/SCHOOL/c++/CSC 2200  /Lab01/Lab01/test1.cpp | |
|  |  |

**Functions**

* + - void [copyTester](#_bookmark69) ([Text](#_bookmark6) copyText)
    - void [print\_help](#_bookmark75) ()

*copyText is passed by value*

* + - int [main](#_bookmark72) ()

### Function Documentation

#### copyTester()

void copyTester (

[Text](#_bookmark6) *copyText* )

Laboratory 1 [test1.cpp](#_bookmark64) Test program for the operations in the [Text](#_bookmark6) ADT Function prototype

**14 File Documentation**

Dummy routine that is passed a text object using call by value. Outputs copyText and clears it. Here is the call graph for this function:



copyTester

Text::clear

Text::showStructure

Here is the caller graph for this function:



main

copyTester

#### main()

int main ( )

Predefined test text objects Destination for assignment Input text object

Input subscript

Character specified by subscript Input test selection

Get user test selection.

Execute the selected test.

Test 1 : Tests the constructors. Test 2 : Tests the length operation.

**4.4 E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/text.cpp File Reference** **15**

Test 3 : Tests the subscript operation.

Test 4 : Tests the assignment and clear operations.

Test 5 : Tests the copy constructor and operator= operations. Here is the call graph for this function:



main

Text::showStructure

print\_help

Text::getLength

copyTester

Text::clear

#### 4.3.1.3 print\_help()

void print\_help ( )

copyText is passed by value

Here is the caller graph for this function:



main

print\_help

## 4.4 E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/text.cpp File Reference

#include *<*iostream*>*

#include *<*iomanip*>*

#include *<*cassert*>*

#include *<*cstring*>*

**16 File Documentation**

#include "Text.h"

Include dependency graph for text.cpp:



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/text.cpp

Text.h

iomanip

cassert

cstring

iostream

stdexcept

**Functions**

* istream & [operator*>>*](#_bookmark83)(istream &input, [Text](#_bookmark6) &inputText)
* ostream & [operator*<<*](#_bookmark81)(ostream &output, const [Text](#_bookmark6) &outputText)

### Function Documentation

#### operator*<<*()

ostream& operator*<<* (

ostream & *output,*

const [Text](#_bookmark6) & *outputText* )

[Text](#_bookmark6) output function. Inserts outputText in ostream output. Returns the state of the output stream.

#### operator*>>*()

istream& operator*>>* (

istream & *input,*

[Text](#_bookmark6) & *inputText* )

[Text](#_bookmark6) input function. Extracts a string from istream input and returns it in inputText. Returns the state of the input stream. Large (but finite)

text buffer

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

Apply the [Text(char )](#_bookmark6) constructor to convert textBuffer to a string. Assign the resulting string to inputText us- ing the assignment operator.

[∗](#_bookmark6)

Return the state of the input stream.

**4.5 E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/Text.h File Reference** **17**

## E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/Text.h File Reference

#include *<*stdexcept*>*

#include *<*iostream*>*

Include dependency graph for Text.h:



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/Text.h

stdexcept

iostream

This graph shows which files directly or indirectly include this file:



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/text.cpp



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/Text.h

E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/lexical.cpp

E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/test1.cpp

**Classes**

* + - class [Text](#_bookmark6)

## E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/textio.cpp File Reference

#include *<*iostream*>*

#include *<*iomanip*>*

**18 File Documentation**

Include dependency graph for textio.cpp:



E:/SCHOOL/c++/CSC 2200

/Lab01/Lab01/textio.cpp

iostream

iomanip

**Functions**

* istream & [operator*>>*](#_bookmark94)(istream &input, [Text](#_bookmark6) &inputText)
* ostream & [operator*<<*](#_bookmark92)(ostream &output, const [Text](#_bookmark6) &outputText)

### Function Documentation

#### operator*<<*()

ostream& operator*<<* (

ostream & *output,*

const [Text](#_bookmark6) & *outputText* )

[Text](#_bookmark6) output function. Inserts outputText in ostream output. Returns the state of the output stream.

#### operator*>>*()

istream& operator*>>* (

istream & *input,*

[Text](#_bookmark6) & *inputText* )

Friends [Text](#_bookmark6) input/output operations (In-lab Exercise 1)

[Text](#_bookmark6) input function. Extracts a string from istream input and returns it in inputText. Returns the state of the input stream. Large (but finite)

text buffer

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

Apply the [Text(char )](#_bookmark6) constructor to convert textBuffer to a string. Assign the resulting string to inputText us- ing the assignment operator.

[∗](#_bookmark6)

Return the state of the input stream.

# Index

Text

∼

Text, [6](#_bookmark7)

buffer

Text, [10](#_bookmark41) bufferSize

Text, [10](#_bookmark41)

clear

Text, [8](#_bookmark22) operator==

Text, [8](#_bookmark22) operator[]

Text, [8](#_bookmark22)

print\_help

test1.cpp, [15](#_bookmark73)

Text, [7](#_bookmark15) config.h

LAB1\_TEST1, [11](#_bookmark51)

LAB1\_TEST2, [11](#_bookmark51)

copyTester

test1.cpp, [13](#_bookmark65)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/Text.h, [17](#_bookmark85)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/config.h, [11](#_bookmark51)

showStructure Text, [9](#_bookmark32)

test1.cpp

copyTester, [13](#_bookmark65)

main, [14](#_bookmark70)

print\_help, [15](#_bookmark73)

Text, [5](#_bookmark5)

∼Text, [6](#_bookmark7)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/lexical.cpp, [12](#_bookmark57)

buffer, [10](#_bookmark41)

bufferSize, [10](#_bookmark41)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/test1.cpp, [13](#_bookmark65)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/text.cpp, [15](#_bookmark73)

E:/SCHOOL/c++/CSC 2200/Lab01/Lab01/textio.cpp, [17](#_bookmark85)

getLength

Text, [7](#_bookmark15)

LAB1\_TEST1

config.h, [11](#_bookmark51) LAB1\_TEST2

config.h, [11](#_bookmark51) lexical.cpp

main, [12](#_bookmark57)

main

lexical.cpp, [12](#_bookmark57)

test1.cpp, [14](#_bookmark70)

operator*<*

Text, [7](#_bookmark15) operator*<<*

Text, [9](#_bookmark32)

text.cpp, [16](#_bookmark78)

textio.cpp, [18](#_bookmark89) operator*>*

Text, [8](#_bookmark22) operator*>>*

Text, [10](#_bookmark41)

text.cpp, [16](#_bookmark78)

textio.cpp, [18](#_bookmark89) operator=

clear, [7](#_bookmark15)

getLength, [7](#_bookmark15)

operator*<*, [7](#_bookmark15)

operator*<<*, [9](#_bookmark32)

operator*>*, [8](#_bookmark22)

operator*>>*, [10](#_bookmark41)

operator=, [8](#_bookmark22)

operator==, [8](#_bookmark22)

operator[], [8](#_bookmark22)

showStructure, [9](#_bookmark32)

Text, [6](#_bookmark7)

toLower, [9](#_bookmark32)

toUpper, [9](#_bookmark32) text.cpp

operator*<<*, [16](#_bookmark78)

operator*>>*, [16](#_bookmark78) textio.cpp

operator*<<*, [18](#_bookmark89)

operator*>>*, [18](#_bookmark89) toLower

Text, [9](#_bookmark32) toUpper

Text, [9](#_bookmark32)

**Chapter 5**

**Attachments**



Laboratory 1: Cover Sheet

Name Razi Khanom/ Phong Nguyen Date 01/25/2018 Section

Place a check mark in the *Assigned* column next to the exercises your instructor has assigned to you. Attach this cover sheet to the front of the packet of materials you submit following the laboratory.

|  |  |  |
| --- | --- | --- |
| **Activities** | **Assigned:** Check or list exercise numbers | **Completed** |
| Implementation Testing | ✓ | ✓ |
| Programming Exercise 1 | ✓ | ✓ |
| Programming Exercise 2 |  |  |
| Programming Exercise 3 |  |  |
| Analysis Exercise 1 | ✓ | ✓ |
| Analysis Exercise 2 |  |  |
|  | Total | 3 |



Laboratory 1: Implementation Testing

Name Razi Khanom/ Phong Nguyen Date 01/25/2018 Section

Check with your instructor whether you are to complete this exercise prior to your lab period or during lab.

Test your implementation of the Text ADT using the program in the file *test1.cpp*. This program supports the following tests.

|  |  |
| --- | --- |
| **Lab 1 Online Test Plans** | |
| **Test** | **Action** |
| 1-1 | Tests the constructors. |
| 1-2 | Tests the length operation. |
| 1-3 | Tests the subscript operation. |
| 1-4 | Tests the assignment and clear operations. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan 1-1 (constructors)** | | | |
| **Test case** | **String** | **Expected result** | **Checked** |
| Simple string Longer string  Single-character string Empty string | alpha epsilon  a  *empty* | alpha epsilon  a  \0 | ✓  ✓  ✓  ✓ |



|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan 1-2 (length operation)** | | | |
| **Test case** | **String** | **Expected length** | **Checked** |
| Simple string Longer string  Single-character string Empty string | alpha epsilon a *empty* | 5  7  1  0 | ✓  ✓  ✓  ✓ |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan 1-3 (subscript operation)** | | | |
| **Test case** | ***n*** | **Expected character** | **Checked** |
| Middle character First character Last character Out of range | 2  0  4  10 | p  a  a  empty | ✓  ✓  ✓  ✓ |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan 1-4 (assignment and clear operations)** | | | |
| **Test case** | **Assignment statement** | **Expected result** | **Checked** |

|  |  |  |  |
| --- | --- | --- | --- |
| Simple assignment Single-character string Empty string  Source string longer than destination buffer  Assign to self  Check assignment by clearing destination | assignStr = alpha; assignStr = a;  assignStr = empty; assignStr = epsilon; assignStr = assignStr;  assignStr = alpha; assignStr.clear(); | alpha  a  \0  epsilon epsilon alpha  \0 | ✓  ✓  ✓  ✓  ✓  ✓  ✓  ✓ |



Laboratory 1: Programming Exercise 1

Name Date Section

|  |  |  |
| --- | --- | --- |
| **Test Plan 1-5 (lexical analysis program)** | | |
| **Test case** | **Expected result** | **Checked** |
| Program in the file *progsamp.dat* | 1. : [void] 2. : [main] | ✓ |
|  | 3 : [(] |  |
|  | 4 : [)] |  |
|  | 5 : [{] |  |
|  | 6 : [int] |  |
|  | 7 : [j] |  |
|  | 8 : [,] |  |
|  | 9 : [total] |  |
|  | 10 : [=] |  |
|  | 11 : [0] |  |
|  | 12 : [;] |  |
|  | 13 : [for] |  |
|  | 14 : [(] |  |
|  | 15 : [j] |  |
|  | 16 : [=] |  |
|  | 17 : [1] |  |
|  | 18 : [;] |  |
|  | 19 : [j] |  |
|  | 20 : [<=] |  |
|  | 21 : [20] |  |
|  | 22 : [;] |  |
|  | 23 : [j] |  |
|  | 24 : [++] |  |
|  | 25 : [)] |  |
|  | 26 : [total] |  |
|  | 27 : [+=] |  |
|  | 28 : [j] |  |
|  | 29 : [;] |  |
|  | 30 : [}] |  |



Laboratory 1: Analysis Exercise 1

Name Date Section

*A full-page version of this exercise with space for writing in answers is available in the online supplements for Lab 1.*

**Part A**

What are the implications of having no destructor in a class like Text that does dynamic memory allocation? What are the practical consequences of not having a destructor for these classes in a long-running program?

**Part B**

What other operators might it make sense to overload in the Text class? Name four and briefly describe how they would work.

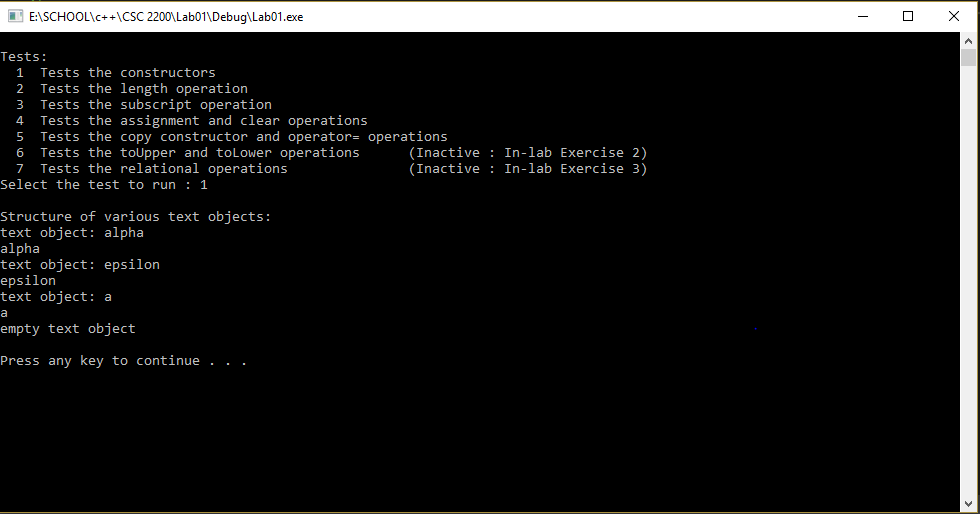
Operator >> (Extraction) and << (insertion), Operator [ ], Operator =

**Part C**

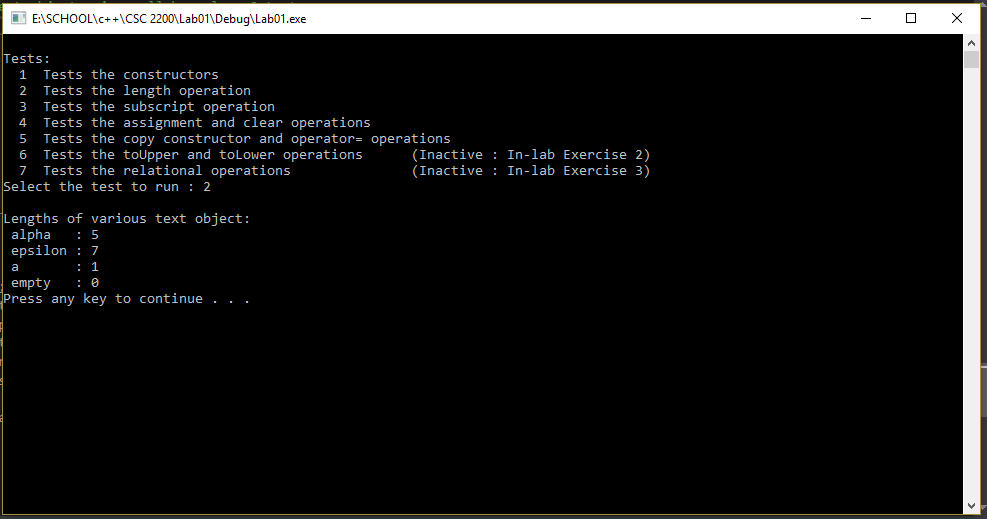
Are there any operators that it does not make sense to overload in the Text class? Why not?

Operator=,<,>, and + are the operators that is not make sense to overload. Because it is useless to compare two text object

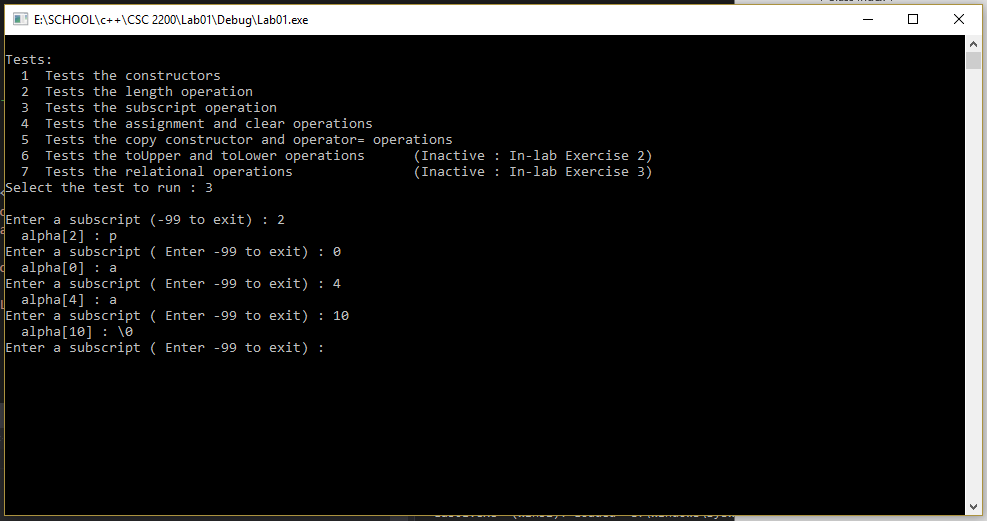
**Test** **Plan 1-1 (constructors)**



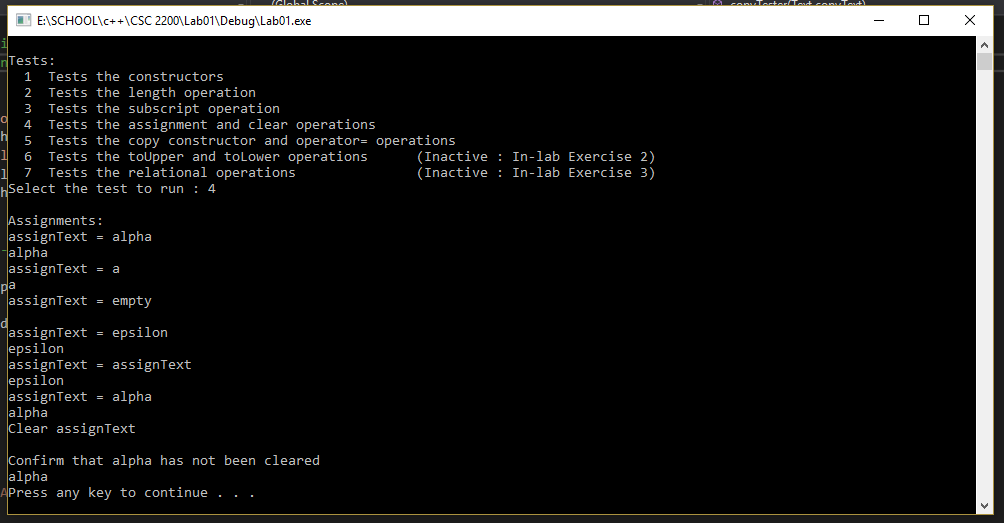
**Test Plan 1-2 (length operation)**



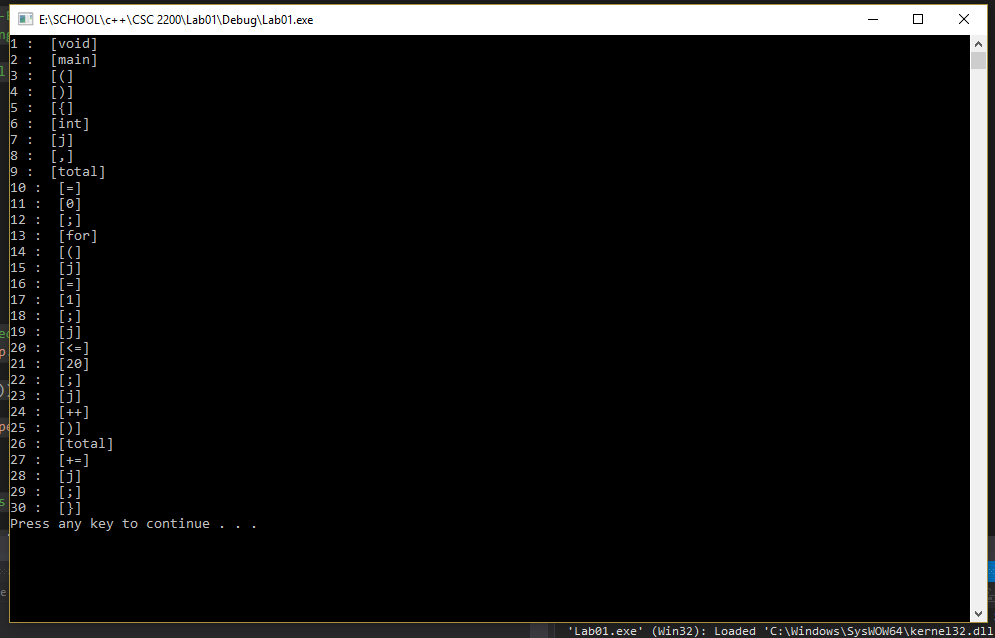
**Test Plan 1-3 (subscri****pt operation)**



**Test Plan 1-4** **(assignment and clear operations)**



**Test Plan 1-5 (lexical analy****sis program)**



**Attachments**

**Te****xt.h**

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

//

//! Laboratory 1 Text.h

//! \*\*Instructor's Solution\*\*

//! Class declaration for the array implementation of the Text ADT

//

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

#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

**Te****xt1.cpp**

//!--------------------------------------------------------------------

//

//! Laboratory 1 test1.cpp

//

//! Test program for the operations in the Text ADT

//

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

#include <iostream>

#include "Text.h"

#include "config.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 )

{

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 (-99 to exit) : ";

cin >> n;

while (n != -99)

{

ch = alpha[n];

cout << " alpha[" << n << "] : ";

if (ch == '\0')

cout << "\\0" << endl;

else

cout << ch << endl;

cout << "Enter a subscript ( Enter -99 to exit) : ";

cin >> n;

}

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.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();

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 : ";

}

**Text.c****pp**

#include <iostream>

#include <iomanip>

#include <cassert>

#include <cstring>

#include "Text.h"

using namespace std;

using namespace std;

Text::Text(const char\* charSeq)

//! Creates a string containing the delimited sequence of characters

//! charSeq. Allocates enough memory for this string.

{

bufferSize = strlen(charSeq) +1 ;

try

{

buffer = new char[bufferSize];

}

catch (bad\_alloc& e)

{

cout << "Allocate Memory Failed" << e.what() << endl;

}

strcpy(buffer, charSeq); //! Copy the string

}

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

Text::Text(const Text& other)

//! Copy constructor

{

bufferSize = other.bufferSize;

buffer = new char[bufferSize];

strcpy(buffer, other.buffer);

}

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

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

//! Assigns other to a Text object.

{

if (other.getLength() > bufferSize) {

delete[] buffer;

bufferSize = other.getLength() + 1;

buffer = new char[bufferSize];

}

strcpy(buffer, other.buffer);

}

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

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

//! Returns the nth character in a Text object -- where the characters are

//! numbered beginning with zero.

{

if (n >= 0 && n <= bufferSize)

return buffer[n];

else

return '\0';

}

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

int Text::getLength()const

//! Returns the number of characters in the Text object buffer (excluding the

//! null character).

{

return strlen(buffer);

}

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

void Text::clear()

//! Clears a Text object -- i.e., makes it empty. The buffer size

//! remains unchanged.

{

buffer[0] = '\0';

}

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

Text::~Text()

//! Frees the memory used by the Text object buffer.

{

delete []buffer;

bufferSize = 0;

}

void Text::showStructure()const

//! Outputs the characters in a string. This operation is intended for

//! testing/debugging purposes only.

{

cout << buffer << endl;

}

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

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;

}

**Lexic****al.cpp**

/\*\*

\* Project Name : Lab 1-Exercise 1

\* Developer Name : Phong Nguyen

\* Date : 01/19/2018

\* Description : Lexical Analysis Program

\*/

#include <fstream>

#include <iostream>

#include "Text.h"

int main()

{

ifstream infile;

//! Token

Text token;

//! Open the specified program file.

infile.open("progsamp.dat");

if (!infile.is\_open())

cout << "Error opening file" << endl;

else

{

//! Counts tokens

int i = 0;

while (infile >> token)

{

i++;

cout << i << " : " << " [" << token << "]" << endl;

}

}

system("pause");

return 0;

}