CSCE 221 Cover Page

Programming Assignment #1

Due **February 1** by midnight to CSNet

First Name: Leilani Last Name: Horlander-Cruz UIN 523008771

User Name leilanihc112 E-mail address leilanihc112@tamu.edu
Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Type of sources	Online	Online	
People			
Web pages (provide URL)	www.cplusplus.com	http:://www.stroustrup.com/Programming	
Printed material			
Other Sources	Powerpoint slides on Data Structures		

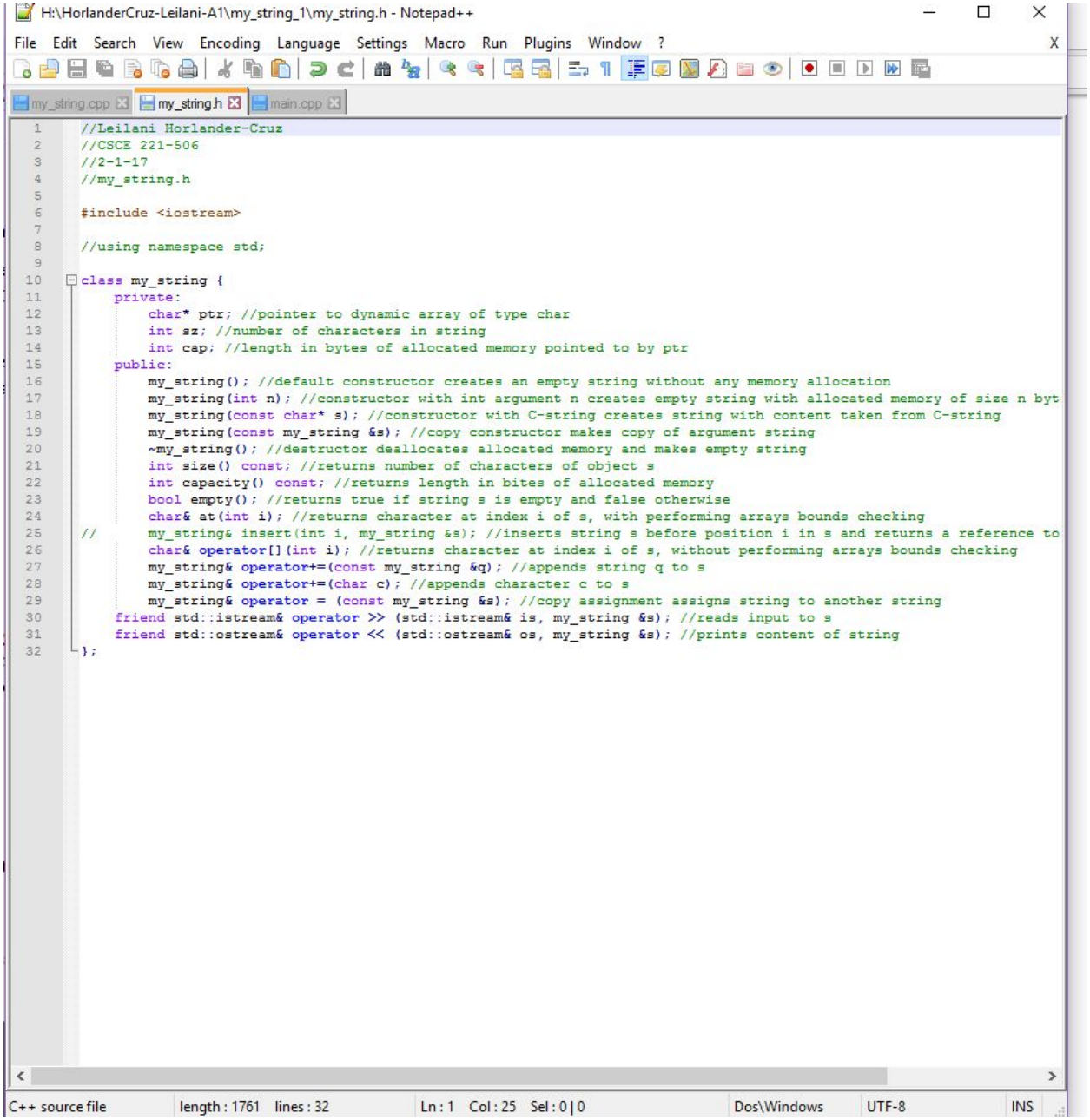
I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work."**On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.**"

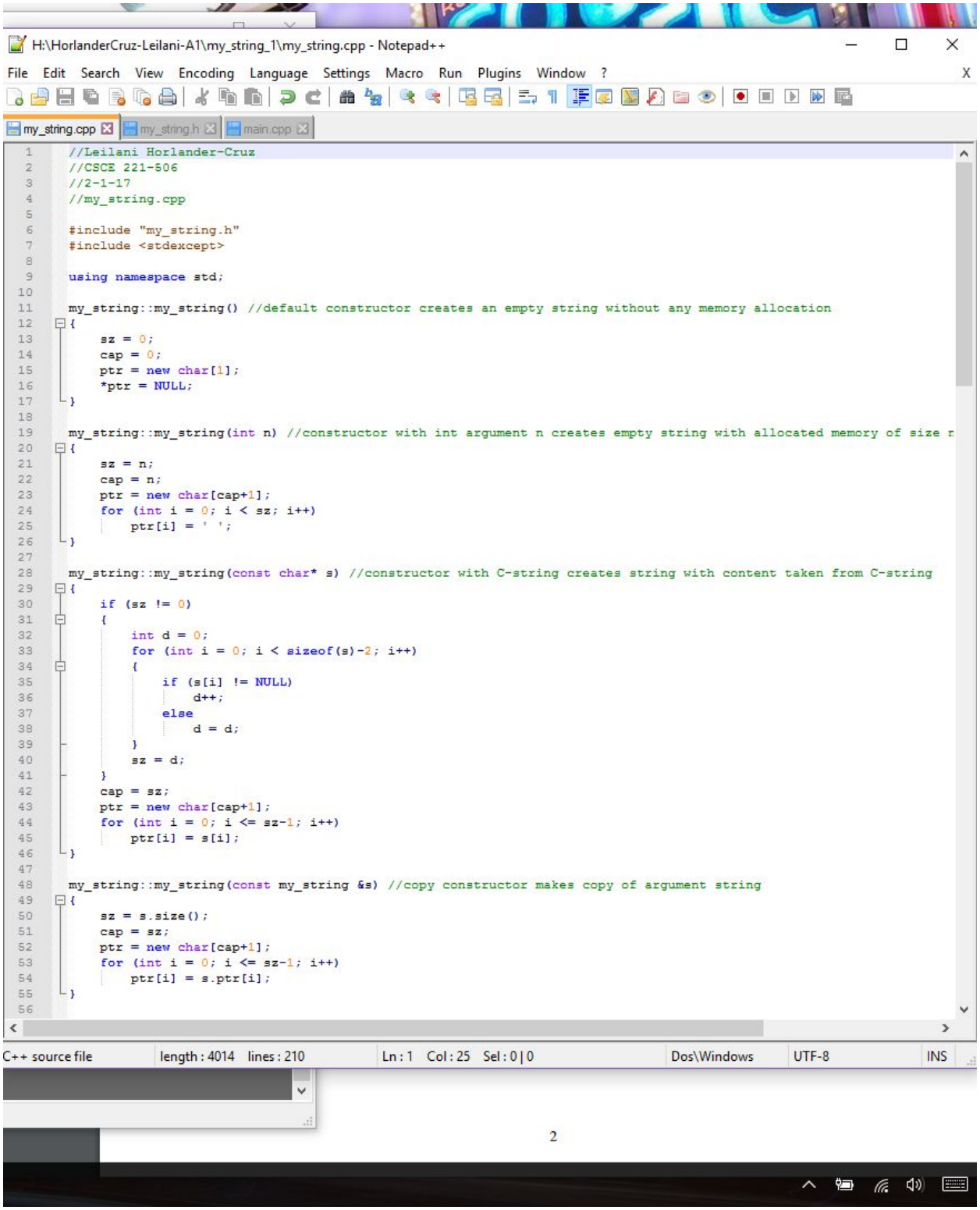
Your Name	Leilani		Horlander-Cruz	Date	2-1-17

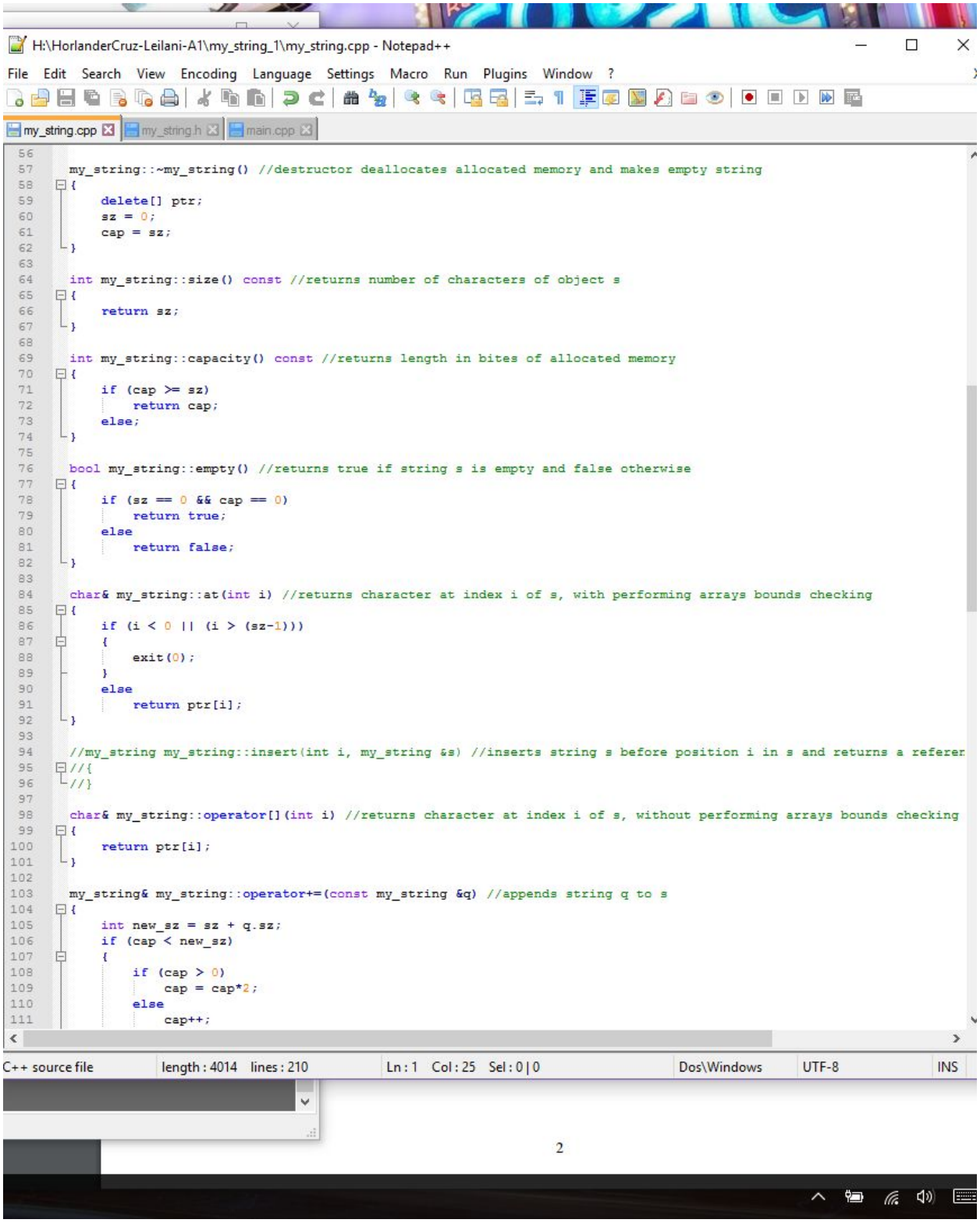
1. This program strives to provide a C++ class called my_string. It is implemented based on a private C++ array of characters that is not fixed in size according to the user's point of view. However, should the user require it, the array is able to reallocate to a larger array. This class does not use the <cstring> library or <string> library already implemented in C++, so it is completely raw.
2. Data Structures and Algorithms
    1. Data structures are ways of organizing and storing data in programming and specifying operations which can be performed on this data. Particularly, this is an array that organizes and stores the data of the string input by the user. An abstract model called Abstract Data Type (ADT) specifies the type of data stored and the operations that support the data.

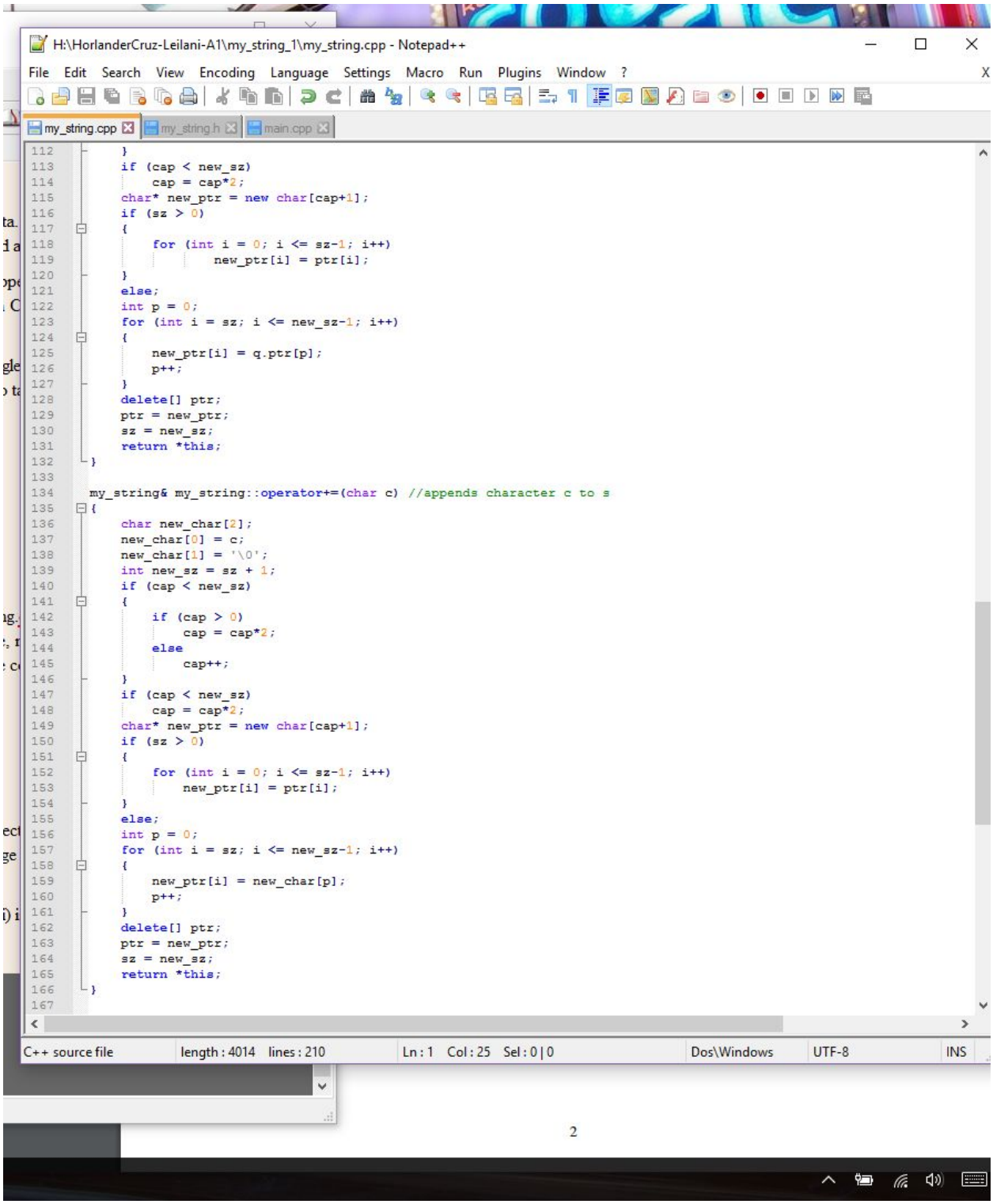
2. The my_string class as a whole is an ADT, and the public member functions such as size() and the overloaded operators to append strings or characters to other strings correspond to the operations of the ADT. The private part of the class, such as sz and cap, depend on a given implementation. In C++, the implementations of different data structures are called containers.
3. In order to solve the assignment problem, the divide and conquer approach was used. This is when attempts to reduce a single large problem into smaller independent subproblems are made. This is a classification of an algorithm. Using different functions and tackling each one little by little in order to tackle the many details that this assignment needed to have satisfied was the approach that worked best in practice.
4. In order to organize and store the data input by the user, an input stream was used, and arrays and pointers of type char were used in order to store this data. An output stream was used in order to print the data back. The capacity was always kept at the amount of elements in the array or more by using logic checks and mathematical calculations.

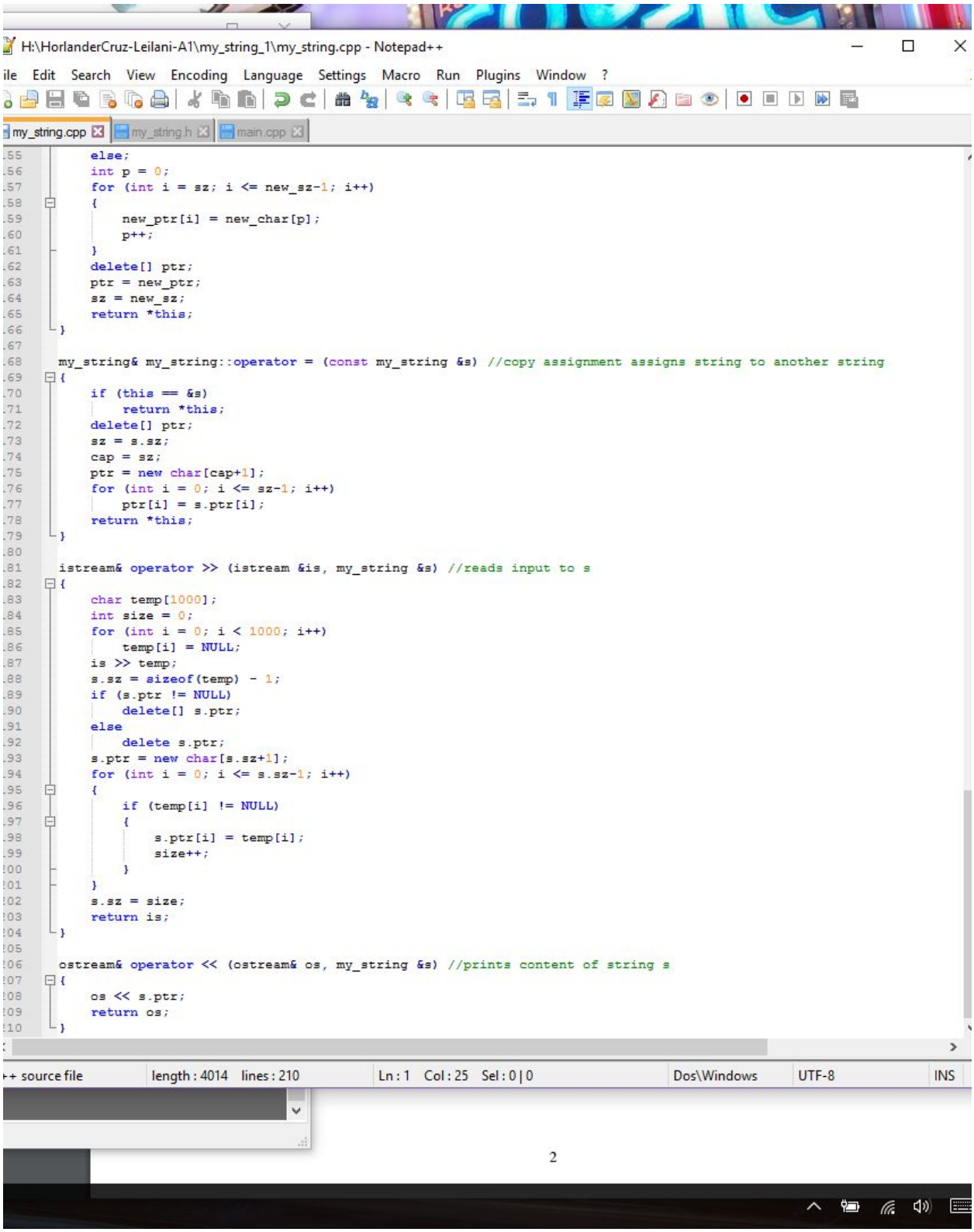
3. C++ organization and implementation of problem solution
    1. The class my_string.cpp is the only class used in this program.
    2.


H:\HorlanderCruz-Leilani-A1\my_string_1\my_string.h - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
my_string.cpp
my_string.h
main.cpp
1 //Leilani Horlander-Cruz
2 //CSCE 221-506
3 //2-1-17
4 //my_string.h
5
6 #include <iostream>
7
8 //using namespace std;
9
10 class my_string {
11 private:
12 char* ptr; //pointer to dynamic array of type char
13 int sz; //number of characters in string
14 int cap; //length in bytes of allocated memory pointed to by ptr
15 public:
16 my_string(); //default constructor creates an empty string without any memory allocation
17 my_string(int n); //constructor with int argument n creates empty string with allocated memory of size n byt
18 my_string(const char* s); //constructor with C-string creates string with content taken from C-string
19 my_string(const my_string &s); //copy constructor makes copy of argument string
20 ~my_string(); //destructor deallocates allocated memory and makes empty string
21 int size() const; //returns number of characters of object s
22 int capacity() const; //returns length in bites of allocated memory
23 bool empty(); //returns true if string s is empty and false otherwise
24 char& at(int i); //returns character at index i of s, with performing arrays bounds checking
25 // my_string& insert(int i, my_string &s); //inserts string s before position i in s and returns a reference to
26 char& operator[](int i); //returns character at index i of s, without performing arrays bounds checking
27 my_string& operator+=(const my_string &q); //appends string q to s
28 my_string& operator+=(char c); //appends character c to s
29 my_string& operator = (const my_string &s); //copy assignment assigns string to another string
30 friend std::istream& operator >> (std::istream& is, my_string &s); //reads input to s
31 friend std::ostream& operator << (std::ostream& os, my_string &s); //prints content of string
32 };
C++ source file
length : 1761 lines : 32
Ln : 1 Col : 25 Sel : 0 | 0
Dos\Windows
UTF-8
INS


H:\HorlanderCruz-Leilani-A1\my_string_1\my_string.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
my_string.cpp
my_string.h
main.cpp
//Leilani Horlander-Cruz
//CSCE 221-506
//2-1-17
//my_string.cpp
#include "my_string.h"
#include <stdexcept>
using namespace std;
my_string::my_string() //default constructor creates an empty string without any memory allocation
{
sz = 0;
cap = 0;
ptr = new char[1];
*ptr = NULL;
}
my_string::my_string(int n) //constructor with int argument n creates empty string with allocated memory of size n
{
sz = n;
cap = n;
ptr = new char[cap+1];
for (int i = 0; i < sz; i++)
ptr[i] = ' ';
}
my_string::my_string(const char* s) //constructor with C-string creates string with content taken from C-string
{
if (sz != 0)
{
int d = 0;
for (int i = 0; i < sizeof(s)-2; i++)
{
if (s[i] != NULL)
d++;
else
d = d;
}
sz = d;
}
cap = sz;
ptr = new char[cap+1];
for (int i = 0; i <= sz-1; i++)
ptr[i] = s[i];
}
my_string::my_string(const my_string &s) //copy constructor makes copy of argument string
{
sz = s.size();
cap = sz;
ptr = new char[cap+1];
for (int i = 0; i <= sz-1; i++)
ptr[i] = s.ptr[i];
}
C++ source file
length : 4014 lines : 210
Ln : 1 Col : 25 Sel : 0 | 0
Dos\Windows
UTF-8
INS


H:\HorlanderCruz-Leilani-A1\my_string_1\my_string.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
my_string.cpp
my_string.h
main.cpp
56
57 my_string::~my_string() //destructor deallocates allocated memory and makes empty string
58 {
59 delete[] ptr;
60 sz = 0;
61 cap = sz;
62 }
63
64 int my_string::size() const //returns number of characters of object s
65 {
66 return sz;
67 }
68
69 int my_string::capacity() const //returns length in bites of allocated memory
70 {
71 if (cap >= sz)
72 return cap;
73 else;
74 }
75
76 bool my_string::empty() //returns true if string s is empty and false otherwise
77 {
78 if (sz == 0 && cap == 0)
79 return true;
80 else
81 return false;
82 }
83
84 char& my_string::at(int i) //returns character at index i of s, with performing arrays bounds checking
85 {
86 if (i < 0 || (i > (sz-1)))
87 {
88 exit(0);
89 }
90 else
91 return ptr[i];
92 }
93
94 //my_string my_string::insert(int i, my_string &s) //inserts string s before position i in s and returns a referer
95 //{
96 //}
97
98 char& my_string::operator[](int i) //returns character at index i of s, without performing arrays bounds checking
99 {
100 return ptr[i];
101 }
102
103 my_string& my_string::operator+=(const my_string &q) //appends string q to s
104 {
105 int new_sz = sz + q.sz;
106 if (cap < new_sz)
107 {
108 if (cap > 0)
109 cap = cap*2;
110 else
111 cap++;
C++ source file
length : 4014 lines : 210
Ln : 1 Col : 25 Sel : 0 | 0
Dos\Windows
UTF-8
INS


H:\HorlanderCruz-Leilani-A1\my_string_1\my_string.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
my_string.cpp
my_string.h
main.cpp
}
if (cap < new_sz)
cap = cap*2;
char* new_ptr = new char[cap+1];
if (sz > 0)
{
for (int i = 0; i <= sz-1; i++)
new_ptr[i] = ptr[i];
}
else;
int p = 0;
for (int i = sz; i <= new_sz-1; i++)
{
new_ptr[i] = q.ptr[p];
p++;
}
delete[] ptr;
ptr = new_ptr;
sz = new_sz;
return *this;
}
my_string& my_string::operator+=(char c) //appends character c to s
{
char new_char[2];
new_char[0] = c;
new_char[1] = '\0';
int new_sz = sz + 1;
if (cap < new_sz)
{
if (cap > 0)
cap = cap*2;
else
cap++;
}
if (cap < new_sz)
cap = cap*2;
char* new_ptr = new char[cap+1];
if (sz > 0)
{
for (int i = 0; i <= sz-1; i++)
new_ptr[i] = ptr[i];
}
else;
int p = 0;
for (int i = sz; i <= new_sz-1; i++)
{
new_ptr[i] = new_char[p];
p++;
}
delete[] ptr;
ptr = new_ptr;
sz = new_sz;
return *this;
}
C++ source file
length : 4014 lines : 210
Ln : 1 Col : 25 Sel : 0 | 0
Dos\Windows
UTF-8
INS



3. There were no features like these used.

4. User guide to compiling and executing the program
    1. In order to compile this program, the user must open the tar file that includes the

header file, my_string.h, and the source file, my_string.cpp. The user should then use Putty in the respective directory, using the command line c++ -std=c++11 *.cpp -o my_string or make all . The main file, main.cpp, can be used in order to run the program, or the user may make their own main file. This file must be in the same directory as the opened tar file before using one of the command lines listed above.
    2. The user should then execute using the command ./my_string .
5. Specifications and descriptions of input and output formats and files.
    1. There are strings input by the user on the input stream.
    2. There is no requirement of input items in this program.
    3. I do not believe there could be an incorrect input.
6. Exceptions
    1. A logical exception was used in the creation of an empty string of size n. Appending a character to the end of it would not work correctly if the elements were of type NULL, so whitespace characters were used instead. Of course, upon improvement and development of this program in the future, a way to change the elements to characters in those spaces could be implemented.
    2. There is one runtime exception included in the C++ file, which is an out_of_range exception. It is thrown if the index in the function at(i) is not in range of the string size (between 0 and size()-1).
7. There was no error when testing with random characters.