Eric Blasko

CSE 330

Winter 2018

Lab 3 – String

* **Status**

100% compete

* **Time Complexity**

The following functions are O(1)

String();

String(String &&);

String & operator=(String &&);

int length();

String(char);

char & operator[](const int);

The following functions are O(n) where n is the size of the string

String(const String &);

String(const char \*);

String & operator=(const String &);

String & operator+=(const String &);

friend bool operator==(const String &, const String &);

friend bool operator<=(const String &, const String &);

friend bool operator<(const String &, const String &);

friend ostream & operator<<(ostream &, const String &);

* **Source Code**

#ifndef STRING\_H

#define STRING\_H

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* 01/24/2018

\* string.h

\* This is the header file for class string. Includes private data member and

\* functions that hold an array of char.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

class String

{

int size;

char \*buffer;

public:

String(); //default constructor

String(const String &); //copy constructor

String(String &&); //move constructor

String(const char \*);

String(char);

~String(); //destructor

int length();

char & operator[](const int);

String & operator=(const String &); //copy assignment

String & operator=(String &&); //move assingment

String & operator+=(const String &);

friend bool operator==(const String &, const String &);

friend bool operator<=(const String &, const String &);

friend bool operator<(const String &, const String &);

friend ostream & operator<<(ostream &, const String &);

};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* 01/24/2018

\* String.cpp

\* This class holds the implementation of constructors and member functions

\* defined in String.h

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <cassert>

#include "String.h"

//default constructor

String::String()

{

size = 0;

buffer = nullptr;

}

//copy constructor

String::String(const String & source)

{

size = source.size;

buffer = new char[size];

assert(buffer != nullptr);

for(int i = 0; i < size; i++)

buffer[i] = source.buffer[i];

}

//move constructor

String::String(String && source): size{source.size}, buffer{source.buffer}

{

source.size = 0;

source.buffer = nullptr;

}

//constructor from "char string" to String

//find size of p, allocate space, assign p to buffer

String::String(const char \*p)

{

size = 0;

for(const char \*q = p; \*q; q++)

size++;

buffer = new char[size];

assert(buffer != nullptr);

for(int i = 0; i < size; i++)

buffer[i] = p[i];

}

String::~String()

{

delete [] buffer;

}

//copy assignment

//similar to char string constructor, buut here

//create a temp buffer based on source,

//delete existing buffer, then assign temp buffer to buffer

String & String::operator=(const String & source)

{

char \*temp = new char[source.size];

assert(temp != nullptr);

for(int i = 0; i < source.size; i++)

temp[i] = source.buffer[i];

delete [] buffer;

buffer = temp;

size = source.size;

return \*this;

}

//move assignment

String & String::operator=(String && source)

{

size = source.size;

source.size = 0;

delete [] buffer;

buffer = source.buffer;

source.buffer = nullptr;

return \*this;

}

bool operator==(const String & left, const String &right)

{

if(left.size != right.size)

return false;

for(int i = 0; i < left.size; i++)

if(left.buffer[i] != right.buffer[i])

return false;

return true;

}

ostream & operator<<(ostream & out, const String & s)

{

for(int i = 0; i < s.size; i++)

out << s.buffer[i];

return out;

}

char & String::operator[](const int pos)

{

return buffer[pos];

}

String::String(char p)

{

size = 1;

buffer = new char(size);

buffer[0] = p;

}

int String::length()

{

return size;

}

//calls two other bool functions in this class

//operator== and operator<, if ether are true

//then statement is true

bool operator<=(const String & left, const String & right)

{

if(left < right || left == right)

return true;

return false;

}

//create a temp buffer the size of current string

//and source. loop twice to copy both strings to

//one string. delete old buffer and assign temp to buffer

String & String::operator+=(const String & source)

{

int tempSize = size + source.size;

char \*temp = new char[tempSize];

assert(temp != nullptr);

for(int i = 0; i < size; i++)

temp[i] = buffer[i];

for(int i = 0; i < tempSize; i++)

temp[size + i] = source.buffer[i];

delete [] buffer;

buffer = temp;

size += source.size;

return \*this;

}

//check if each array char is equal to eachother.

//return boolean when difference is found

bool operator<(const String & left, const String & right)

{

int n = 0;

while(left.buffer[n] == right.buffer[n])

n++;

if(left.buffer[n] < right.buffer[n])

return true;

else

return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* test1.cpp

\* 01/31/2018

\* This program test the String.h functions. If assert catchs an error it

\* will end the program. If success is printed to screen then implentation

\* of String class is functioning properly

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cassert>

#include "String.h"

using namespace std;

int main()

{

String s1;

assert(s1.length() == 0);

String s2{"hi"};

assert(s2.length() == 2);

String s3{s2};

assert(s3.length() == 2);

assert(s3[0] == 'h');

assert(s3[1] == 'i');

s1 = s2;

assert(s1 == s2);

s3 = "bye";

assert(s3.length() == 3);

assert(s3[0] == 'b');

assert(s3[1] == 'y');

assert(s3[2] == 'e');

s1 += "re";

assert(s1 == "hire");

s1 += "d";

assert(not (s1 == "hire"));

cout << "SUCCESS" << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* test2.cpp

\* 01/31/2018

\* This program test the functionality of the class String.h. It will

\* test the overload functions for that class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include "String.h"

using namespace std;

int main()

{

String s1{"abcd"}, s2{"abe"};

if (s1 == s2)

cout << s1 << " == " << s2;

else

if (s1 <= s2)

cout << s1 << " < " << s2 << endl;

else

cout << s1 << " > " << s2 << endl;

s1 = move(s2);

cout << "s1=" << s1 << " s2=" << s2 << endl;

String s3 = move(s1);

cout << "s1=" << s1 << " s3=" << s3 << endl;

}

* **Sample Runs**

**Test1.cpp**

Script started on 2018-01-30 22:39:58-0800

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ g\_\_^H^[[K^H^[[K\_+^H^[[K^H^[[K++ -c stri^H^[[K^H^[[K^H^[[K^H^[[KString.cpp^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ g\_^H^[[K++ -c S^H^[[Ktest1.cpp^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ g++ String.o St^H^[[K^H^[[Ktest1.o^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ ./a.out^M

SUCCESS^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ exit^M

Script done on 2018-01-30 22:40:48-0800

**Test2.cpp**

Script started on 2018-01-30 22:40:56-0800

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ g++ -c String.cpp^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ g++ -c test2.cpp^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ g++ String.o test2.o^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ ./a.out^M

abcd < abe^M

s1=abe s2=^M

s1= s3=abe^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab3^G[005670557@csusb.edu@jb358-1 lab3]$ exit^M

Script done on 2018-01-30 22:41:28-0800