

CS 202 - Assignment 6 - Spring 2022

Mr. Piotrowski

March 2022



1 Introduction

This assignment should look familiar! This is the part II of our strings assignment (*which was assignment 4*). Here, you will practice overloading operators for the same class.

If you successfully completed some functions in Assignment 4 (listed below), you will be able to rely on them heavily in this assignment. By using these functions, you can avoid a lot of pointer manipulation!

1. Concatenate(const CS202_String&, const CS202_String&)
2. FindSubString(const CS202_String&)
3. DeleteAt(int)

4. At(int)
5. GetSize()

2 Getting Started

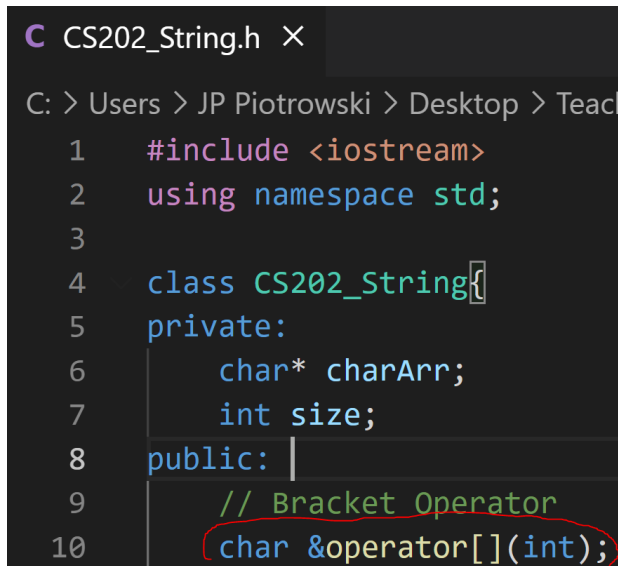
NOTE: The word "string" cannot appear in your code **AT ALL**, even in comments. If a single instance of "string" is found, codegrade will give you 0 points.

You will have to add **16 Operator Overloads** to your class. I have provided the code for one of the overloads for you; it is in a text file called **ExtractionOp.txt**. The remaining 15 operators you will have to overload. Please add the operator overload prototypes to your class in CS202.String.h and fill out their definitions in the CS202.String.cpp file.

Download the new Main.cpp file and use this for the assignment. Also download the **StringFile.txt**. This file is opened during the program.

2.1 Member Operator Overloads

The following operator must be overloaded as a class member.



```
C CS202_String.h X
C: > Users > JP Piotrowski > Desktop > Teach
1  #include <iostream>
2  using namespace std;
3
4  class CS202_String{
5  private:
6      char* charArr;
7      int size;
8  public:
9      // Bracket Operator
10     char &operator[](int);
```

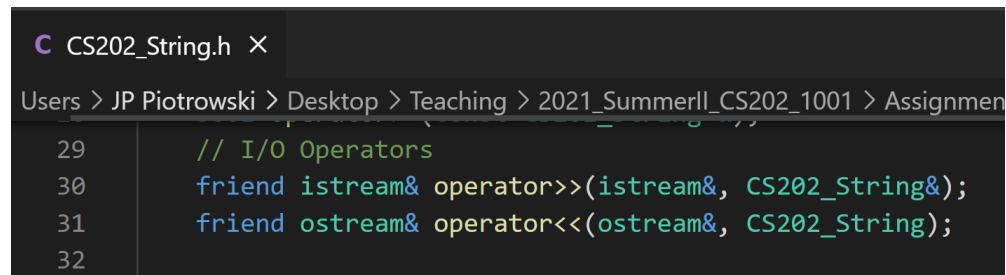
2.1.1 Bracket Operator

The bracket operator should return the character within the string located at the index. The index is the int parameter passed to the operator. The At(int) function works almost exactly like the bracket operator except for one thing:

the bracket operator returns the actual reference to the character. `At(int)` just returns a copy. Because the actual reference is returned, if the index is out of bounds, it is best to just call `exit(1)`; which will terminate the program.

2.2 Global Operator Overloads

The following operators must be overloaded as a global operators.



```
C CS202_String.h X
Users > JP Piotrowski > Desktop > Teaching > 2021_SummerII_CS202_1001 > Assignment
29 // I/O Operators
30 friend istream& operator>>(istream&, CS202_String&);
31 friend ostream& operator<<(ostream&, CS202_String);
32
```

2.2.1 Extraction Operator

At this time, download the **ExtractionOp.txt** file and copy-paste this code in your `CS202_String.cpp` file. Also, don't forget to add the friend portion to the `CS202_String.h` file. This operator will take a string of characters from a stream (`cin`, `ifstream`, etc...) and overwrite the `CS202_String` on the RHS of the expression. For example:

```
1 CS202_String myString("Beans");
2 cin >> myString;
```

Whatever the user inputs for `cin` will overwrite the string "Beans" and store the new input into `myString`. This operator can be rather complicated for string classes and that is why I gave it to you.

2.2.2 Insertion Operator

This is the operator that is called when you use:

```
1 CS202_String myString("Beans");
2 cout << myString;
```

You can check the class code matrix code for an idea of how this one looks. It is a very simple operator in which you will insert each individual character from the `CS202_String` into the `ostream&` parameter. You will use a for loop. Once all characters have been sent to the `ostream`, return the `ostream`.

2.3 Operator Overloads of Your Choice

The following operators can be overloaded as either global operators or class members. They are shown here as class members. If you choose to make them global, you will need to make these operators friends of the class and add a CS_202 parameter to each operator.

```
11 // Addition operators
12 CS202_String operator+(CS202_String);
13 CS202_String operator+(const char*);
14 // ++ Operators
15 CS202_String operator++(); // Pre op
16 CS202_String operator++(int); // Post op
17 // Subtraction Operators
18 CS202_String operator-(CS202_String);
19 CS202_String operator-(const char*);
20 // Multiplication operators
21 CS202_String operator*(unsigned int);
22 // Boolean Operators
23 bool operator<(const CS202_String &);
24 bool operator==(const CS202_String &);
25 bool operator!=(const CS202_String &);
26 bool operator>(const CS202_String &);
27 bool operator<=(const CS202_String &);
28 bool operator>=(const CS202_String &);
```

Please do not be intimidated. Most of these operators can easily be done by using all of the other functions from Assignment 4. **I would recommend completing them in this order. You can call operators within definitions of other operators!**

2.3.1 Addition Operators

There are 2 addition operators "+". One operator takes in a CS202_String and the other one takes in a character array. The job of the addition operators is to concatenate the 2 strings and return a brand new string. You should not make any changes to either of the operands. For example:

```
1 CS202_String s1("Beans"), s2("Rice");
2 CS202_String s3 = s1 + s2;
3 // or
4 CS202_String s1("Beans");
5 CS202_String s3 = s1 + "Rice";
```

The s3 object now contains "BeansRice". Remember, you already have the Concatenate function. Each of these operators can be done in one line of code. For the overload that takes a character array as a parameter, you could call the parameter constructor to convert it into a CS202_String.

2.3.2 PlusPlus Operators

There are 2 PlusPlus operators "++". One is the pre-op and the other is the post-op. The pre-op will make a copy of the first character and add it to the beginning of the string. For example:

```
1 CS202_String s1("Beans");
2 cout << ++s1 << endl;
3 cout << s1 << endl;
```

The "++s1" now makes the string "BBeans". It will also return a copy of the string with the copied character, so the output of the above code is:

```
1 BBeans
2 BBeans
```

The post-op does the same thing as the pre-op, except to the last character in the string. So the following code works like this:

```
1 CS202_String s1("Beans");
2 cout << s1++ << endl;
3 cout << s1 << endl;
```

The "s1++" now makes the string "Beanss". It will also return a copy of the string with the copied character, so the output of the above code is:

```
1 Beanss
2 Beanss
```

2.4 The Subtraction Operators

There are 2 subtraction operators "-". One operator takes in a CS202_String and the other one takes in a character array. The job of the subtraction operators is to remove the RHS string from the LHS string and return a brand new string. You should not make any changes to either of the operands. If the substring cannot be found, just return a copy of the LHS object. For example:

```
1 CS202_String s1("Beans"), s2("ea");
2 CS202_String s3 = s1 - s2;
3 // or
4 CS202_String s1("Beans");
5 CS202_String s3 = s1 - "ea";
```

The s3 object now contains "Bns". Remember, you already have the FindSubString function and DeleteAt function. Each of these operators can be done in a few lines of code. For the overload that takes a character array as a parameter, you could call the parameter constructor to convert it into a CS202_String.

2.4.1 The Multiplication Operator

There is 1 multiplication operator "*". This operator takes in an int parameter (say 3) and copies the string that many times (3 times) and return a brand new string. You should not make any changes to either of the operands. For example:

```
1 CS202_String s1("Beans");
2 s1 = s1 * 3;
```

The s1 object now contains "BeansBeansBeans". Remember, you already have the concatenate function. This function could be as simple as using a for loop and concatenating a string over and over again.

2.5 The Boolean Operators

I highly recommend doing the LessThan and EqualEqual operators first. The other operators can be coded using these first 2 operators.

2.5.1 Less Than Operator

There is one less than operator "<". This operator returns true if the LHS is alphabetically less than the RHS. I only expect you to compare the ascii values of the chars and nothing more than that. Do not worry about converting the strings to lower or upper case.

2.5.2 Equality Operator

There is one equality operator "==". This operator returns true if the LHS is identical to the RHS. I only expect you to compare the ascii values of the chars and nothing more than that. Do not worry about converting the strings to lower

or upper case. Also, if the 2 strings being compared are the same object, print this message:

```
1 "Duh! This is the same CS202_string obviously. Just look at
  their addresses!"
```

This message will print when the following code happens:

```
1 CS202_String s1("Beans");
2 if(s1 == s1){
3     // some code
4 }
```

How can you tell if 2 things are the same object? Well check the address of course!

2.5.3 The Does Not Equal Operator

There is one DNE operator "!=". This operator returns true if the 2 strings are not exactly the same. This operator can be coded in one line by calling the equality operator and using "!".

2.5.4 Greater Than Operator

There is one greater than operator ">". This operator returns true if the LHS string is alphabetically greater than the RHS string. This can be done in one line of code by using "!" with the less than operator and the equality operator.

2.5.5 Greater Than or Equal Operator

There is one greater than equal operator ">=". This operator returns true if the LHS string is alphabetically greater than the RHS string, or if they are the same string. This can be done in one line of code by using the greater than operator and the equality operator.

2.5.6 Less Than or Equal Operator

There is one less than equal operator "<=". This operator returns true if the LHS string is alphabetically lesser than the RHS string, or if they are the same string. This can be done in one line of code by using the less than operator and the equality operator.

3 Compiling Your Code

Compile your code using

```
1 g++ -std=c++11 -g Main.cpp CS202_String.cpp
```

Run your code using

```
1 ./a.out
```

Use valgrind!

```
1 valgrind --leak-check=full ./a.out
```

4 Comprehension Questions

Please answer the following questions. Your answers should be short (1 line!).

1. What is the reason that some operators must be members of the class? Which operators in the CS202_String class have to be class members?
2. What is the difference in the operator prototypes for the pre ++ and the post ++?
3. Why do you think overloading operators might be helpful? Feel free to google this one.

5 Submission

Submit to codegrade.