Aufgabe 4: String Klasse mit überladenen Operatoren (C++)

Implementierungsaufgabe. Teile der Aufgabe sind vorgegeben. Siehe Anhang "Sourcefiles".

String Klasse mit überladenen Operatoren

Erstellen Sie eine String-Klasse ohne zu Hilfenahme von STL, die es Ihnen erlaubt, den folgenden Code zu schreiben:

```
int main( ) {
  String s1;
  String s2("Hello");
  String s3(s2);
  s1 += s2; s2 = s3;
  cout << s2 << endl;
  cout << s2[ 2 ] << endl;
};
```

Beachte

- Intern soll der String als ein Zeiger auf ein Zeichen dargestellt werden (wie in C).
- Bei der Konstruktion, Kopieren und bei der Zuweisung soll der String dupliziert werden. Sie benötigen daher einen Kopierkonstruktor, Zuweisungsoperator und einen Destruktor
- Folgende Operaten sollen überladen werden: `+=` (Konkatenation), `=` (Zuweisung) und [] (Arrayzugriff)

Tips und Hinweise

Für den Arrayzugriff `string[index]` (sprich die überladene Methode) verwenden Sie bitte die folgende Signatur.

```
char& operator[](int index)
```

Wieso wird eine Referenz auf zurückgegeben? Beachte dass eine Arrayzugriff auch auf der linken Seite einer Zuweisung vorkommen kann.

Unten finden Sie ein mögliches Grundgerüst.

```
#include <iostream>
using namespace std;


class String {
 private:
  // 'String' is represented internally as a plain C-style string.
  int size;
  char* str;
 public:
  String() {
    size = 0;
    str = new char[1];
    str[0] = '\0';
  }
  String(char c) {
    size = 1;
    str = new char[2];
    str[0] = c;
    str[1] = '\0';
  }
  ~String() { delete[] str; }

  // make friend, so we can access private members
  friend ostream& operator<< (ostream &out, String &s);

};

ostream& operator<< (ostream &out, String &s) {
  for(int i=0; i<s.size; i++) {
    out << s.str[i];
  }

  return out;
}


int main() {
  String s;
  String s2('H');

  cout << s << endl;
  cout << s2 << endl;

}
```

Sourcefiles

Verwenden Sie folgende Sourcefiles.

```
// String.h
#include <iostream>
using namespace std;

class String {
 private:
```

```
  // 'String' is represented internally as a plain C-style string.
  int size;
  char* str;
 public:
  String();
  String(char c);
  String(const char *);
  String(const String&);
  ~String();

  char& operator[](int index);
  String& operator=(String&);
  String& operator+=(String&);

  // make friend, so we can access private members
  friend ostream& operator<< (ostream &out, String &s);

};


// String.cpp
#include <iostream>
using namespace std;

#include "String.h"


String::String() {
    size = 0;
    str = new char[1];
    str[0] = '\0';
}


String::String(char c) {
    size = 1;
    str = new char[2];
    str[0] = c;
    str[1] = '\0';
}

String::String(const char *s) {
  // TODO
}

String::String(const String& s) {
  // TODO
}

String::~String() { delete[] str; }

char& String::operator[](int index) {
  // TODO
}

String& String::operator=(String& s) {
  // TODO
}

String& String::operator+=(String& s) {
  // TODO
}


#include "String.h"
```

```
ostream& operator<< (ostream &out, String &s) {
  for(int i=0; i<s.size; i++) {
    out << s.str[i];
  }

  return out;
}


int main() {
  String s;
  String s2('H');

  cout << s << endl;
  cout << s2 << endl;

}
```