TPP2020-HW4-40847025S 蕭瀜 （請記得修改這裡！未填學號與姓名將會扣很大！若為多人作業，請列出所有成員。）

```
/* ------------------------------------------------------------
// String.h
// ---------------------------------------------------------- */
#ifndef __MYSTRING_H__
#define __MYSTRING_H__

typedef unsigned long size_t;
class String {
public:
	String();
	String(const String&);
	String(const char*);
	~String() { delete str_; };
	size_t size() const;
	size_t capacity() const;
	const char* c_str() const;
	const char& operator[](size_t) const;
	char& operator [] (size_t);
	String& operator = (const String&);
	String& operator = (const char*);
	String& operator = (char);
	String& operator += (const String&);
	String& operator += (const char*);
	String& operator += (char);
	String operator + (const String&);
	String operator + (const char*);
	String operator + (const char);
	void clear();
	void swap(String&);
	friend bool operator == (const String&, const String&);
	friend bool operator == (const char*, const String&);
	friend bool operator == (const String&, const char*);
	friend bool operator != (const String&, const String&);
	friend bool operator != (const char*, const String&);
	friend bool operator != (const String&, const char*);
	friend bool operator < (const String&, const String&);
	friend bool operator < (const char*, const String&);
	friend bool operator < (const String&, const char*);
	friend bool operator <= (const String&, const String&);
	friend bool operator <= (const char*, const String&);
	friend bool operator <= (const String&, const char*);
	friend bool operator > (const String&, const String&);
	friend bool operator > (const char*, const String&);
	friend bool operator > (const String&, const char*);
	friend bool operator >= (const String&, const String&);
	friend bool operator >= (const char*, const String&);
	friend bool operator >= (const String&, const char*);
	friend std::ostream& operator << (std::ostream&, const String&);
	friend std::istream& operator >> (std::istream&, String&);
private:
	char *str_ = nullptr;
	size_t size_ = 0, capacity_ = 0;
};

#endif


/* ------------------------------------------------------------
```

TPP2020-HW4-40847025S 蕭瀜 （請記得修改這裡！未填學號與姓名將會扣很大！若為多人作業，請列出所有成員。）

```
// String.cpp
// -------------------------------------------------------------- */
#include <iostream>
#include <cstring>
#include <cstdlib>
#include "String.h"

String::String():size_(0), capacity_(15), str_(new char[15]) {}
String::String(const String &s):size_(s.size_), capacity_(s.capacity_), str_(new char[s.capacity_]) {
        strcpy(str_, s.str_);
}
String::String(const char *s):size_(strlen(s)), capacity_((size_ > 15) ? size_ : 15) {
        str_ = new char[capacity_];
        strcpy(str_, s);
}
size_t String::size() const { return size_; }
size_t String::capacity() const { return capacity_;}
const char* String::c_str() const { return str_; }
const char& String::operator[](size_t i) const { return str_[i]; }
char& String::operator [] (size_t i) { const_cast<char &> (static_cast<const String &>(*this)[i]);}
String& String::operator = (const String& s) {
        size_ = s.size_;
        if(capacity_ < s.capacity_) {
                capacity_ = s.capacity_;
                delete str_;
                str_ = new char [capacity_];
        }
        strcpy(str_, s.str_);
        return *this;
}
String& String::operator = (const char* s) {
        size_ = strlen(s);
        if(capacity_ < size_) {
                capacity_ = size_;
                delete str_;
                str_ = new char [capacity_];
        }
        strcpy(str_, s);
        return *this;
}
```

```
String& String::operator = (char c) {
	char s[] = " ";
	s[0] = c;
	strcpy(str_, s);
	return *this;
}
String& String::operator += (const String& s) {
	size_ += s.size_;
	if(size_ > capacity_) {
		capacity_ <<= 1;
		str_ = (char *)realloc(str_, capacity_);
	}
	strcat(str_, s.str_);
	return *this;
}
String& String::operator += (const char* s) {
	size_ += strlen(s);
	if(size_ > capacity_) {
		capacity_ <<= 1;
		str_ = (char *)realloc(str_, capacity_);
	}
	strcat(str_, s);
	return *this;
}
String& String::operator += (char c) {
	size_ += 1;
	if(size_ > capacity_) {
		capacity_ <<= 1;
		str_ = (char*)realloc(str_, capacity_);
	}
	str_[size_ - 1] = c;
	return *this;
}
String String::operator + (const String& s) {
	String tmp(*this);
	return tmp += s;
}
String String::operator + (const char* s) {
	String tmp(*this);
```

```
    return tmp += s;
}
String String::operator + (char s) {
    String tmp(*this);
    return tmp += s;
}
void String::clear() {
    size_ = 0;
    str_[0] = 0;
}
void String::swap(String& s) {
    String tmp;
    tmp.size_ = size_;
    tmp.capacity_ = capacity_;
    tmp.str_ = str_;
    size_ = s.size_;
    capacity_ = s.capacity_;
    str_ = s.str_;
    s.size_ = tmp.size_;
    s.capacity_ = tmp.capacity_;
    s.str_ = tmp.str_;
    tmp.str_ = nullptr;
}
bool operator == (const String& lhs, const String& rhs) { return strcmp(lhs.str_, rhs.str_) == 0;}
bool operator == (const char* lhs, const String& rhs) { return strcmp(lhs, rhs.str_) == 0;}
bool operator == (const String& lhs, const char* rhs) { return strcmp(lhs.str_, rhs) == 0;}
bool operator != (const String& lhs, const String& rhs) { return strcmp(lhs.str_, rhs.str_) != 0;}
bool operator != (const char* lhs, const String& rhs) { return strcmp(lhs, rhs.str_) != 0;}
bool operator != (const String& lhs, const char* rhs) { return strcmp(lhs.str_, rhs) != 0;}
bool operator < (const String& lhs, const String& rhs) { return strcmp(lhs.str_, rhs.str_) < 0;}
bool operator < (const char* lhs, const String& rhs) { return strcmp(lhs, rhs.str_) < 0;}
bool operator < (const String& lhs, const char* rhs) { return strcmp(lhs.str_, rhs) < 0;}
bool operator <= (const String& lhs, const String& rhs) { return strcmp(lhs.str_, rhs.str_) <= 0;}
bool operator <= (const char* lhs, const String& rhs) { return strcmp(lhs, rhs.str_) <= 0;}
bool operator <= (const String& lhs, const char* rhs) { return strcmp(lhs.str_, rhs) <= 0;}
bool operator > (const String& lhs, const String& rhs) { return strcmp(lhs.str_, rhs.str_) > 0;}
bool operator > (const char* lhs, const String& rhs) { return strcmp(lhs, rhs.str_) > 0;}
bool operator > (const String& lhs, const char* rhs) { return strcmp(lhs.str_, rhs) > 0;}
bool operator >= (const String& lhs, const String& rhs) { return strcmp(lhs.str_, rhs.str_) >= 0;}
bool operator >= (const char* lhs, const String& rhs) { return strcmp(lhs, rhs.str_) >= 0;}
bool operator >= (const String& lhs, const char* rhs) { return strcmp(lhs.str_, rhs) >= 0;}
std::ostream& operator << (std::ostream& os, const String& s) {
	os << s.c_str();
	return os;
}
std::istream& operator >> (std::istream& is, String& s) {
	char ch = is.get();
	while(ch == ' ' || ch == '\n') ch = is.get();
	while(ch != ' ' && ch != '\n' && ch != EOF) {
		s += ch;
		ch = is.get();
	}
	return is;
}


/* ------------------------------------------------------------
// main.cpp
// ---------------------------------------------------------- */
#include <iostream>
#include "String.h"

using namespace std;

void Constructors() {
	cout << "Constructors:" << endl;
	String s, t("CSIE108"), u(t);
	cout << s << endl;
	cout << t << endl;
	cout << u << endl;
	cout << endl;
}

void Copy() {
	cout << "Copy assignment:" << endl;
	String s1("NTU"), t1("NTNU");
	s1 = t1;
	cout << s1 << endl;
	cout << t1 << endl;
	cout << endl;
}

void Size_C_str() {
	cout << "size() and c_str():" << endl;
	String s("NTNU");
	cout << s.size() << endl; // 4
	const char *p = s.c_str();
	cout << p << endl;
	cout << endl;
}

void ReadWrite(String &str) {
	cout << "ReadWrite ";
	str[2] = 'T';
	cout << str << endl;
}
void ReadOnly(const String &str) {
	cout << "ReadOnly ";
	char c = str[1];
	cout << c << endl;
}
void RW() {
	cout << "operator []: " << endl;
	String s("NTNU");
	ReadWrite(s);
	ReadOnly(s);
	cout << endl;
}

void PlusE() {
	cout << "operator +=:" << endl;
	String s("NT"), t("NU");
	s += t;
	cout << s << endl;
	cout << endl;
}

void Plus() {
        cout << "operator +:" << endl;
        String s("NT"), t("NU"), u;
        u = s + t;
        cout << u << endl;
        cout << endl;
}

void Clear() {
        cout << "clear():" << endl;
        String s("NT");
        s.clear();
        cout << s << endl;
        cout << endl;
}

void Swap() {
        cout << "swap():" << endl;
        String s("NTNU"), u("NTU");
        cout << "before: " << s << "," << u << endl;
        s.swap(u);
        cout << "after: " << s << "," << u << endl;
        cout << endl;
}

int main() {
        Constructors();
        Copy();
        Size_C_str();
        RW();
        PlusE();
        Plus();
        Clear();
        Swap();
}
```

<< 請適當編排以利列印與閱讀，程式碼儘量不要跨行。 >>