作业：实现一个自定义 String 类

> 如果在网页上查看公式有问题，请查看随附的 `README.pdf` 文件。

作业概述

在本次作业中，你需要设计并实现一个基于 `char` 数组的 `string` 类。**禁止使用 C 和 C++ 的标准库 `string` 类 以及其他 STL 容器**。目标是构建一个简化版本的类似 `std::string` 字符串类，实现基本的字符串操作、内存管理、运算符重载等功能。

String 类介绍

在 C++ 中，`string` 是用来表示字符序列的类。它在处理字符串的长度、拼接、插入等操作时提供了极大的便利。你将在本作业中重新实现类似的功能。此 `string` 类需要能够高效地管理动态内存并提供一系列成员函数用于操作字符串。

基本功能介绍

1. 默认构造函数

该构造函数应初始化一个空字符串，并设置一个合理的初始容量（例如 16）。字符串的容量是其内部缓冲区的大小，而长度是字符串当前包含的字符数。

```
String();  // 构造空字符串
```

2. 根据字面量构造字符串

你需要实现一个可以接受 `const char*` 字面量的构造函数，创建一个对应的字符串对象。

```
String(const char* str);  // 使用C风格的字符串初始化
```

3. 复制构造函数

实现一个复制构造函数，允许通过现有的 `String` 对象构造一个新的 `String`。

```
String(const String& other);  // 拷贝构造
```

4. 移动构造函数

实现移动构造函数，允许在无需复制数据的情况下将数据从一个 `String` 对象移动到另一个。

```
String(String&& other) noexcept;  // 移动构造
```

5. 复制赋值运算符

你需要重载赋值运算符，使得可以将一个 `String` 对象的值复制给另一个 `String` 对象。

```
String& operator=(const String& other);  // 复制赋值
```

6. 移动赋值运算符

实现移动赋值运算符，避免不必要的数据复制。

```
String& operator=(String&& other) noexcept;  // 移动赋值
```

7. 运算符 `+`

你需要重载 `+` 运算符，使两个字符串可以通过此运算符进行拼接。

```
String operator+(const String& other) const;  // 字符串拼接
```

8. 运算符 `[]`

重载 `[]` 运算符，允许通过索引访问字符串中的字符。

```
char& operator[](size_t index);  // 返回字符引用
const char& operator[](size_t index) const;  // 常量字符串访问
```

9. 成员函数 `size()` 和 `capacity()`

- `size()`：返回字符串的当前长度。
- `capacity()`：返回字符串的内部缓冲区容量。

```
size_t size() const;  // 返回字符串长度
size_t capacity() const;  // 返回字符串容量
```

10. 成员函数 `insert()`

实现一个插入函数，允许在指定位置插入另一个字符串。

```
void insert(size_t pos, const String& str);  // 在指定位置插入字符串
```

11. 输出流运算符重载 `<<` 和 输入流运算符重载 `>>`

你需要重载 `<<` 和 `>>` 运算符，支持将 `String` 对象输出到输出流以及从输入流中读取字符串。

```
friend std::ostream& operator<<(std::ostream& os, const String& str);  // 输出流重载
friend std::istream& operator>>(std::istream& is, String& str);  // 输入流重载
```

12. 类型转换

实现一个类型转换函数，允许将 `String` 对象转换为 `const char*` 类型。

```
operator const char*() const;  // 转换为const char*
```

可选功能（选做部分）

1. RC

不依赖 `shared_ptr` 实现一个引用计数机制，允许多个 `String` 对象共享相同的数据，减少不必要的复制操作。不要求线程安全。

```
String(const String& other);  // 拷贝构造，不复制数据
String& operator=(const String& other);  // 复制赋值，不复制数据
String clone() const;  // 返回一个新的String对象，完全复制当前对象的数据
```

设计要求

1. 内存管理

你需要正确管理字符串的内存，确保所有动态分配的内存都能在合适的时机被释放，避免内存泄漏。

- 字符串的容量应根据需要自动增长。
- 当字符串长度达到其容量时，应重新分配内存并将现有数据复制到新缓冲区中。

2. 函数接口要求

所有成员函数和重载运算符应根据设计的接口实现。

测试用例

你需要自行编写一组测试用例，验证以下功能是否正确：

- 字符串的构造、赋值、移动、拼接。
- 字符串长度和容量的正确维护。
- 插入操作的正确性。
- 运算符重载的正确性。

提交方式

- 将代码打包成一个 zip 文件，文件名格式为：`姓名-学号-Assignment2.zip`。
- 将文件发送至邮箱：`submit@vollate.top`，邮件标题为：`ARTINX2025视觉Assignment2`

> 请将 `姓名` 和 `学号` 替换为你的姓名和学号。务必保证邮件标题完全一致，否则可能无法正常提交作业。