自定义string函数

1- string 赋值运算符举例

String::opreator=函数的优化：

最一般的写法，特点：使用const string& 传参防止临时对象。

```
String& String::operator =(const String & rhs)
{
    if (itsString)
        delete [] itsString;
    itsLen = rhs.GetLen();
    itsString = new char[itsLen+1];
    for (unsigned short i = 0;i<itsLen;i++)      /// 一个个的拷贝
        itsString[i] = rhs[i];
    itsString[itsLen] = '/0';
    return *this;
}
```

a. 优化1，防止自我间接赋值， a = b; c = b; a = c; 如果没有第一个if判断，当把c赋给a的时候，删除了a.itsString，后面的拷贝就会出错。注意是**if(this==&rhs)**, 而不是**if(*this==rhs)** .

```
String& String::operator =(const String & rhs)
{
    if (this == &rhs)    /// 判断地址，防止自己赋值给自己
        return *this;
    if (itsString)
        delete [] itsString;
    itsLen=rhs.GetLen();
    itsString = new char[itsLen+1];
    for (unsigned short i = 0;i<itsLen;i++)
        itsString[i] = rhs[i];
    itsString[itsLen] = '/0';
    return *this;
}
```

b. 优化2，不进行拷贝赋值，只是交换控制信息，而且是强异常安全：

构建一个临时对象，然后交换。之后临时对象自动销毁

```
String & String::operator = (String const &rhs)
{
    if (this != &rhs)
        String(rhs).swap (*this); // Copy-constructor and non-throwing swap

    // Old resources are released with the destruction of the temporary
above
    return *this;
}
```

c. 优化3，以最原始的传值方式传参，避免临时对象创建：

这时候临时对象是在传参的时候创建的。

```
String & operator = (String s) // the pass-by-value parameter serves as a
temporary
{
   s.swap (*this); // Non-throwing swap
   return *this;
}// Old resources released when destructor of s is called.
```

d. copy and swap 的右值优化，详见https://en.wikibooks.org/wiki/More_C++_Idioms/Copy-and-swap

2- 网络版string

```
#include <iostream>
#include <cstring>
using namespace std;
class String
{
        public:
                String();
                String(const char *const);
                String(const String &);
                ~String();
                char & operator[] (unsigned short offset);
                char operator[] (unsigned short offset)const;
                String operator+(const String&);
                void operator+=(const String&);
                String & operator= (const String &);
                unsigned short GetLen()const {return itsLen;}
                const char * GetString()const {return itsString;}
        private:
                String (unsigned short);
                char * itsString;
                unsigned short itsLen;
};
String::String()
{
        itsString = new char[1]; //为什么设置成1，这样会导致内存1bytes无法释放吗?
我觉得和itsString = new char没区别，那他为什么要设置成1，这样有什么用? 21天学会C++那
本书，我也有 ，书上也确实是设置成1.
        itsString[0] = '/0';
        itsLen=0;
}
String::String(unsigned short len)
{
        itsString = new char[len+1];
        for (unsigned short i =0;i<=len;i++)
                itsString[i] = '/0';
        itsLen=len;
}
String::String(const char * const cString)
{
        itsLen = strlen(cString);
        itsString = new char[itsLen+1];
        for (unsigned short i=0;i<itsLen;i++)
                itsString[i] = cString[i];
        itsString[itsLen] = '/0';
}
String::String(const String & rhs)
{
        itsLen = rhs.GetLen();
        itsString = new char[itsLen+1];
        for (unsigned short i = 0;i<itsLen;i++)
                itsString[i] = rhs[i];
        itsString[itsLen] = '/0';
}
String::~String()
{
        delete [] itsString;
        itsLen = 0;
}
String& String::operator =(const String & rhs)
{
        if (this == &rhs)
                return *this;
        delete [] itsString;
        itsLen=rhs.GetLen();
        itsString = new char[itsLen+1];
        for (unsigned short i = 0;i<itsLen;i++)
                itsString[i] = rhs[i];
        itsString[itsLen] = '/0';
        return *this;
}
char & String::operator [](unsigned short offset) //这个程序这样写，起到了什么用处？？和main中的那一个对应？
{
        if (offset > itsLen)
                return itsString[itsLen-1]; //这个返回itslen-1到底是什么意思？为什么要减去1 ？？
        else
                return itsString[offset];
}
char String::operator [](unsigned short offset)const
{
        if (offset > itsLen)
                itsString[itsLen-1];
        else
                return itsString[offset];
}
String String::operator +(const String& rhs)
{
        unsigned short totalLen = itsLen + rhs.GetLen();
        String temp(totalLen);
        unsigned short i;
        for (i=0;i<itsLen;i++)
                temp[i] = itsString[i];
        for (unsigned short j = 0;j<rhs.GetLen();j++,i++)
                temp[i] = rhs[j];
        temp[totalLen] = '/0';
        return temp;
}
void String::operator +=(const String& rhs)
{
        unsigned short rhsLen = rhs.GetLen();
        unsigned short totalLen = itsLen + rhsLen;
        String temp(totalLen);
        unsigned short i;
        for (i = 0;i<itsLen;i++)
                temp[i] = itsString[i];
        for (unsigned short j = 0;j<rhs.GetLen();j++,i++)
                temp[i] = rhs[i-itsLen];
        temp[totalLen] = '/0';
}
int main()
{

        String s1("initial test"); //调用了什么函数?
        cout<<"S1:/t"<<s1.GetString()<<endl;
        char *temp ="Hello World";
        s1 = temp;//调用了什么函数?
        cout<<"S1:/t"<<s1.GetString()<<endl;
        char tempTwo[20];
        strcpy(tempTwo,"; nice to be here!");
        s1 += tempTwo;
        cout<<"tempTwo:/t"<<tempTwo<<endl;
        cout<<"S1:/t"<<s1.GetString()<<endl;
        cout<<"S1[4]:/t"<<s1[4]<<endl;
        cout<<"S1[999]:/t"<<s1[999]<<endl;//调用了什么函数?
        String s2(" Anoter string");//调用了什么函数?
        String s3;
        s3 = s1+s2;
        cout<<"S3:/t" <<s3.GetString()<<endl;
        String s4;
        s4 = "Why does this work?";//调用了什么函数?
        cout<<"S4:/t"<<s4.GetString()<<endl;
        return 0;
}
```