补充：

5、 TString 类的定义与实现

5.1. 引入

字符串的基本操作都是基于一维字符数组的，因此其赋值、比较、连接、求长度等方法均要用函数去实现，且实现过程中必须注意空间是否足够、最后是不是有'\0'等情况。

5.2. 实现目标

参照 string 类的方法，用比较简洁易懂的方法来实现字符串的基本操作。

5.3. 要求实现的基本操作

5.3.1. 定义对象并初始化：

```
① TString s1;                    //s1 为 NULL
② TString s1("hello");           //s1 为"Hello"
③ TString s1="hello";            //s1 为"Hello"
④ TString s1("Hello"), s2=s1;    //s1/s2 均为"Hello"
⑤ char *s = "Hello";
   TString s1 = s;               //s1 为"Hello"
⑥ char s[] = "Hello";
   TString s1 = s;               //s1 为"Hello"
```

5.3.2. 输入操作：（以空格/回车做为输入结束）

```
① TString s1;
   cin >> s1;   //若键盘输入 Hello,则 s1 得到"Hello"
② TString s1;
   cin >> s1;   //若键盘输入 Hello 123,则 s1 得到"Hello"(空格为分隔符)
```

5.3.3. 输出操作：

```
① TString s1("hello");
   cout << s1;   //输出"hello"
② TString s1;
   cout << s1;   //输出"<NULL>"，说明：这是对 NULL 进行的特殊处理
```

5.3.4. 取字符串操作：（将 TString 中的内容以 char *方式返回，只读，不可更改内容）

```
① TSTring s1("hello");
   printf("%s", s1.c_str());     //输出为"hello"
```

5.3.5. 赋值操作：

```
① TString s1("hello"), s2;
   s2=s1;         //s2 为"Hello"
② TString s1("Hello");
   s1="Hi";       //s1 为"Hi",原"Hello"不再保留
```

5.3.6. 连接操作：（运算符+表示字符串连接后赋值给另一个串）

```
① TString s1("tong"), s2("ji"), s3;
   s3 = s1+s2;     //s3 为"tongji"
   s3 = s2+s1;     //s3 为"jitong"
② TString s1("tong"), s3;
   s3 = s1+"ji";   //s3 为"tongji"
   s3 = "ji"+s1;   //s3 为"jitong"
③ TString s1("tong"), s3;
   char *s="ji";
   s3 = s1+s;      //s3 为"tongji"
   s3 = s+s1;      //s3 为"jitong"
```

④ TString s1("tong"), s3;
```
char s[]="ji";
s3 = s1+s;      //s3为"tongji"
s3 = s+s1;      //s3为"jitong"
```
⑤ TString s1("Hello"), s3
```
char c = '!';
s3 = s1 + c;              //s3为"Hello!"
```
⑥ TString s1("ello"), s3
```
s3 = 'H' + s1;            //s3为"Hello"
```

5.3.7. 自连接操作：（运算符+=表示字符串连接后赋值给自己）

① TString s1("tong"), s2("ji");
```
s1 += s2;       //s1为"tongji"
```
② TString s1("tong");
```
s1 += "ji";     //s1为"tongji"
```
③ TString s1("tong");
```
char *s="ji";
s1 += s;        //s1为"tongji"
```
④ TString s1("tong");
```
char s[]="ji";
s1 += s;        //s1为"tongji"
```
⑤ TString s1("Hello");
```
char c = '!';
s1 += c;        //s1为"Hello!"
```

5.3.8. 删除操作：（运算符-表示从字符串中删除另一个字符串/一个字符后赋值给另一个串）

① TString s1("tongji"), s2("ji"), s3;
```
s3 = s1 - s2;         //s3为"tong"
```
② TString s1("tongji"), s3;
```
s3 = s1 - "ji";       //s3为"tong"
```
③ TString s1("tongji"), s3;
```
char *s="ji";
s3 = s1 - s;          //s3为"tong"
```
④ TString s1("tongji"), s3;
```
char s[]="ji";
s3 = s1 - s;          //s3为"tong"
```
⑤ TString s1("tongji"), s3;
```
char c1 = 'j', c2 = 'i';
s3 = s1 - c1 - c2;    //s3为"tong"
```

5.3.9. 自删除操作：（运算符-=表示从字符串中删除另一个字符串/一个字符后赋值给自己）

① TString s1("tongji"), s2("ji");
```
s1 -= s2;         //s1为"tong"
```
② TString s1("tongji");
```
s1 -= "ji";       //s1为"tong"
```
③ TString s1("tongji");
```
char *s="ji";
s1 -= s;          //s1为"tong"
```
④ TString s1("tongji");
```
char s[]="ji";
s1 -= s;          //s1为"tong"
```

⑤ TString s1("tongji");
char c1 = 'j', c2 = 'i';
(s1 -= c1) -= c2; //s1 为"tong"

5.3.10. 复制操作：（运算符*表示将字符串自身复制若干倍后赋值给另一个串）

① TString s1("tong"), s2;
s2 = s1*2; //s2 为"tongtong"

② TString s1, s2;
s2 = s1*5; //s2 为<NULL> "

5.3.11. 自复制操作：（运算符*表示将字符串自身复制若干倍后赋值给自己）

① TString s1("tong"), s2;
s1 *= 2; //s1 为"tongtong"

② TString s1;
s1 *= 5; //s1 为<NULL> "

5.3.12. 反转操作：（运算符!表示将字符串反转后赋值给另一个串）

① TString s1("tong"), s2;
s2 = !s1; //s2 为"gnot"，s1 仍为"tong"

② TString s1;
s2 = !s1; //s2 为<NULL> "

5.3.13. 比较操作：（按 strcmp 的规则返回即可）

① TString s1="house", s2="horse";
s1 > s2; (包括其它 5 种比较运算) //返回 0/1

② TString s1="house";
s1 > "horse"; (包括其它 5 种比较运算) //返回 0/1
"horse" > s2; (包括其它 5 种比较运算) //返回 0/1

③ TString s1="house";
char *s="horse"
s1 > s; (包括其它 5 种比较运算) //返回 0/1
s > s2; (包括其它 5 种比较运算) //返回 0/1

④ TString s1="house";
char s[]="horse"
s1 > s; (包括其它 5 种比较运算) //返回 0/1
s > s2; (包括其它 5 种比较运算) //返回 0/1

5.3.14. 求串长度：（按 strlen 的规则返回即可）

① TString s1("Hello");
cout << s1.length(); //输出为 5

② 定义全局函数 TStringLen(const TString &);
TString s1("Hello"), s2("123");
char *s3="abcde";
char s4[]="wxyz";
TStringLen(s1+s2); //返回值为 8
TStringLen(s2+s1); //返回值为 8
TStringLen(s1+"pq"); //返回值为 7
TStringLen("pq"+s1); //返回值为 7
TStringLen(s1+s3); //返回值为 10
TStringLen(s3+s1); //返回值为 10
TStringLen(s1+s4); //返回值为 9
TStringLen(s4+s1); //返回值为 9

5.3.15.　取串中某个字符的值/给串中的某个字符赋值：（按字符数组的规则即可）

① TString s1("hello");
　cout << s1[1];　　//输出为 e

② TString s1("hello");
　s1[0] -= 32;
　cout << s1;　　//输出为 Hello

【要求：】1、程序由三个文件组成，各文件的说明如下：

14-b5.h：给出 TString 类的定义及其它需要的定义

14-b5.cpp：给出 TString 类的所有成员函数的实现及其它需要的全局函数的实现

14-b5-main.cpp：在 main 函数中给出了 TString 类的测试用例，不准修改，不需要提交，检查作业时会替换本文件

2、在操作系统的内存允许的情况下，**均不再考虑空间是否够用，但也不能浪费空间**

例 1：TString s1("Hello"); 则最多允许申请 6 个字节的空间，**不能多申请**

例 2：TString s1; 若 s1 = ***** 或 s1 = s1 + ***** 等语句反复出现时，不能简单限定空间不超过多少字节，**要无尽利用空间，直到内存空间被耗尽为止**

例 3：TString s1("Hello"); 若 s1 = s1-"He";则 s1 占用空间要减为 4 字节

5、实现过程**不允许**使用系统提供的 string 类，但可以使用<cstring>中字符串函数

6、最后的 100MB 累加测试完成后，用“任务管理器”查看的内存占用**不能超过 115MB**

7、最后的 100MB 累加测试，+=方式的完成时间，**不允许**超过+方式的 60%

8、给出 Windows 下的 14-b5-demo.exe 供参考

9、给出 Linux 下的 14-b5-demo 供参考（在$下输入 14-b5-demo 即可运行，不需要./）

```
s1已有长度：     99.896 MB字节，本次增加 63029 字节
s1已有长度：    99.9402 MB字节，本次增加 46300 字节
s1已有长度：    99.9985 MB字节，本次增加 61173 字节
s1已有长度：    100.048 MB字节，本次增加 52112 字节，总用时    248.906秒，本次1MB用时      5.156秒
time=248.906
内存分配到达满100MB，测试结束
    本次测试耗时 248.906秒
内存性能测试(s1=s1+str方式)
    老师的机器（CPU：AMD Ryzen 7 4700U，内存：DDR4 3200MHz）运行VS2019-Debug-x86编译的程序，大约耗时240-260秒
    【说明】：只有相同编译器下的运行时间才有可比性
              如果时间相差太大，除CPU和内存的性能差异外，还有可能是算法问题

观察任务管理器中本程序的内存占用情况(不允许超过115MB)...

按回车键继续
```

```
s1已有长度：    99.8798 ■B字节，本次增加 40464 字节
s1已有长度：    99.9207 ■B字节，本次增加 42893 字节
s1已有长度：    99.9556 ■B字节，本次增加 36574 字节
s1已有长度：    100.002 ■B字节，本次增加 49203 字节，总用时    114.391秒，本次1■B用时      2.06秒
time=114.391
内存分配到达满100■B，测试结束
    本次测试耗时 114.391秒
内存性能测试(s1=s1+str方式)
    Linux服务器下运行，大约耗时110-130秒（如果多人同时测试，偏差可能较大）
    【说明】：只有相同编译器下的运行时间才有可比性
              如果时间相差太大，除CPU和内存的性能差异外，还有可能是算法问题

按回车键继续
```

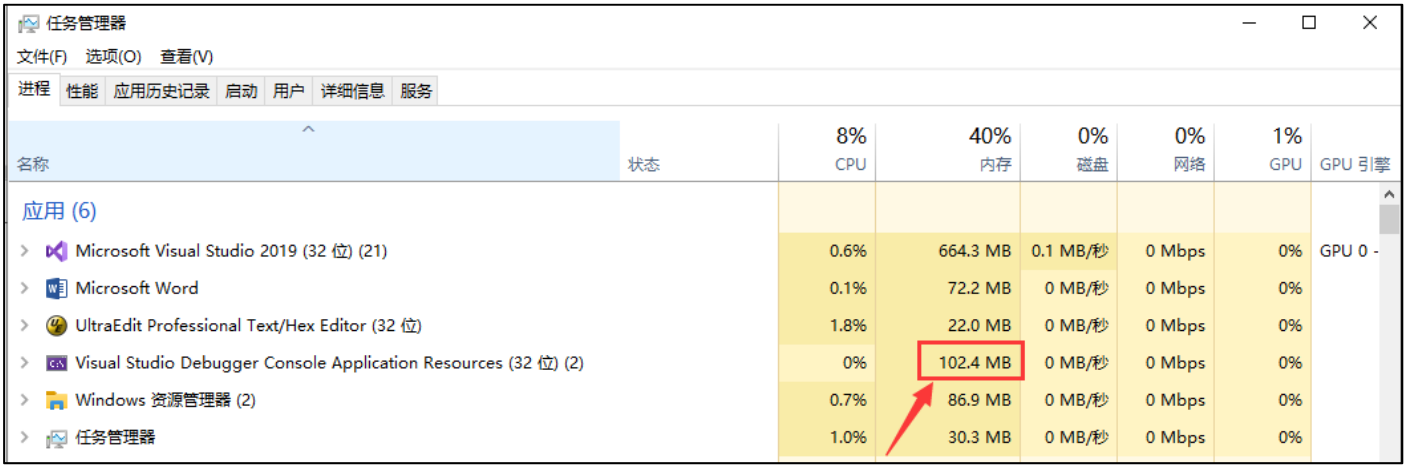
```
s1已有长度:     99.8582 MB字节，本次增加 56765 字节
s1已有长度:     99.9194 MB字节，本次增加 64135 字节
s1已有长度:     99.9582 MB字节，本次增加 40649 字节
s1已有长度:     100.009 MB字节，本次增加 52925 字节，总用时    112.953秒，本次1MB用时      2.093秒
time=112.953
内存分配到达满100MB，测试结束
    本次测试耗时 112.953秒
内存性能测试(s1+=str方式)
    老师的机器（CPU：AMD Ryzen 7 4700U，内存：DDR4 3200MHz）运行VS2019-Debug-x86编译的程序，大约耗时100-120秒
    【说明】：只有相同编译器下的运行时间才有可比性
              如果时间相差太大，除CPU和内存的性能差异外，还有可能是算法问题

观察任务管理器中本程序的内存占用情况(不允许超过115MB)...

按回车键继续
```

```
s1已有长度:     99.8587 MB字节，本次增加 56650 字节
s1已有长度:     99.9115 MB字节，本次增加 55420 字节
s1已有长度:     99.9692 MB字节，本次增加 60505 字节
s1已有长度:     100.012 MB字节，本次增加 45014 字节，总用时     51.016秒，本次1MB用时      1.011秒
time=51.017
内存分配到达满100MB，测试结束
    本次测试耗时 51.017秒
内存性能测试(s1+=str方式)
    Linux服务器下运行，大约耗时50-70秒（如果多人同时测试，偏差可能较大）
    【说明】：只有相同编译器下的运行时间才有可比性
              如果时间相差太大，除CPU和内存的性能差异外，还有可能是算法问题

按回车键继续
```



【编译器要求：】

		编译器VS	编译器Dev	编译器Linux
14-b5.h	TString类-头文件	Y	Y	Y
14-b5.cpp	TString类-实现	Y	Y	Y

【作业要求：】

1、5月6日前网上提交本次作业
2、每题所占平时成绩的具体分值见网页
3、超过截止时间提交作业则不得分