StringBuffer

ver 2.0

В езика C начинът, по който се реализират низове е масив от символи, който завършва с 0. Но така програмите, които работят с низове, се организират около кода, а не около данните (самите низовете).

Задача:

Да се реализира клас **StringBuffer.** Класът трябва да представя низ, на който могат да бъдат променяни стойността и дължината. Капацитетът на низа се променя така, че във всеки момент **StringBuffer** има капацитет по-голям от действителния низ, който представя (операцията **Append** може да се осъществява много ефективно).

Конструктори
StringBuffer()
StringBuffer(char * str)
StringBuffer(int capacity)
StringBuffer(char * str, int capacity)
StringBuffer(StringBuffer & stringBuffer)
Деструктор

Методи
int Length()
int Capacity()
int IndexOf(char character) * // -1, ако не е намерен
char CharAt(int position) * // ако е дадена невалидна позиция 0
bool Equals(StringBuffer & comparedString)
int CompareTo(StringBuffer & comparedString) // както в C
void Append(char * stringToAppend)
void Append(StringBuffer & stringToAppend)
void Append(char c)
void Print() // Този метод ще бъде заменен в последствие
StringBuffer& Substring(int start, int end) *

Методи
//от start включително до символа преди end

* **Позицията започва от 0.**

За да бъде реалистичен този клас има нужда от още методи. Например в JAVA има още **replace, reverse,** 12 **insert** и т.н.

Класът, който реализирате, **трябва да се подчинява на този интерфейс**. Вие сами трябва да прецените всичките **private** данни и методи, който ви трябват. Помислете преди да започнете да пишете!

1.1 Като използвате написания клас, напишете програма **reverse.cpp**, която обръща низ въведен от потребителя. StringBuffer Reverse(StringBuffer b).

1.2 Като използвате написания клас, напишете програма, която заменя във **n** въведени от потребителя низа (**n** също се въвежда) тага **
** с **\n** и ги извежда на екрана.

Оператори
==
!=
<
>
<=
>=
+ // Слепва два низа. StringBuffer b1, b2, b3 • b1=b2+b3; • b1="some string here"+b2; **(1)** • b1=b2+"some string here"; **(2)** Каква е разликата между **(1)** и **(2)**?
=
+=
~ // reverse.

2.1 Изчислете:

StringBuffer b();

StringBuffer b1("*Bjarne Stroustrup*");

StringBuffer b2("In C++ it's harder to shoot yourself in the foot, but when you do, you blow off your whole leg");

```
b=b2+“\n“+b1;
b1+=~b2;
if(b1<=b2) cout<<“b1 <= b2“;
```

Използвайте шаблони, за да може класа StringBuffer да работи както със символи от тип **char** така и с **wchar_t.** Последният тип не трябва да ви притеснява. Това е **wide character,** който е 16 битов за разлика от 8 битовия **char**. Очевидно е, че с 16 бита може да се кодират много повече символи.

Забележете използването на L в долния пример:

```
wchar_t wc;
wc = L'B';
wchar_t* p = L"Hi";
```

На функциите, които знаете съответстват същите, но **str** е заменено с **wsc**. Пример:

strcpy	**wcscpy**
strcmp	**wcscmp**

Също:

cin	**wcin**
cout	**wcout**

По подразбиране StringBuffer е низ от **char.**

Условието ще бъде допълвано! Пазете това, което сте писали.