Subject: Re: StringBuffer size [BUG]
Posted by piotr5 on Tue, 26 May 2009 12:44:05 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 19 May 2009 07:27But that would defeat the purpose of StringBuffer. You are using the constructor variant with len parameter mostly if you need to "convert" external data to String, for example in LoadFile.
what does this mean? how do I call the constructor of StringBuffer at the exact location where my data is already stored in another container? if it were stored in a string-buffer then I could just pick this, then I don't need a constructor with len-parameter. are there any examples of re-interpreting some memory-position as a string-buffer of certain size? I really do not understand why StringBuffer actually needs a constructor with length-parameter!

a related idea, a possibility which is missing in U++. it is something similar to the array-type in c. we have vector and similar containers which basically are just pointers to some data along with the necessary beaurocrazy. what I am missing is a class which I could just sort-of reinterpret_cast to and from char[]. I imagine a templated class like:

```
template <class T,int L> class StaticArray
{...
T t[L];
...};
StaticArray<int,5> primes={{2,3,5,7,11}};
```

with all the pick-constructors, conversions and stuff. is there any library providing such a basic storage-class? I can imagine this would provide some opportunity for range-checking, and if the class T would provide a constructor alike T(0), then even a length could be defined as either zero-terminated or in case of zero-less StaticArray as L. I think in some badly written book on C I read that a "char s[50]" passed as a parameter to string-functions will be treated as zero-terminated, and in case of no zero it will default to the size 50. I've never seen this to work for C but it would be a nice feature for U++. is anyone interested in that kind of thing? but I guess I'm just curious how a pick-constructor could be implemented here. just imagine:

```
StaticArray<char,11> f1() {
StaticArray<char,11> out={"hello world"};
return out;
}

main() {
StaticArray<char,11> h=f1();
h[4]=' ';
Cout()<<f1()<<"\n"; //output: "hell  world"
}
```