Subject: Re: String improvements
Posted by mirek on Sat, 25 Feb 2006 21:59:07 GMT
View Forum Message <> Reply to Message

hojtsy wrote on Sat, 25 February 2006 12:12I added a feature to String::Find and String::ReverseFind that position could be also given relative to the end of the String, with -1 being equal to the length.

```
s.Find('R', -3) == s.Find('R', s.GetCount()+1-3);
s.ReverseFind('R', -3) == s.ReverseFind('R', s.GetCount()+1-3);
```

Implementation is below:template <class T, class S>

```
int AString<T, S>::Find(int chr, int from) const
{
 ASSERT(from >= -GetLength()-1 && from <= GetLength());
 if(from < 0)
  from += GetLength()+1;
 const T *e = End();
 for(const T *s = ptr + from; s < e; s++)
  if(*s == chr)
   return s - ptr;
 return -1;
}

template <class T, class S>
int AString<T, S>::ReverseFind(int chr, int from) const
{
 ASSERT(from >= -GetLength()-1 && from <= GetLength());
 if(from < 0)
  from += GetLength()+1;
 const T *s = ptr + from;
 while(--s >= ptr)
  if(*s == chr)
   return s - ptr;
 return -1;
}
```

This makes the AString::ReverseFind(int chr) unnecessary, because the ReverseFind(int chr, int from) could have -1 as default value for "from". Could you please add this improvement to uppsrc?

I will have think about it a little. It seems as logical extension, however there is one thing I do not exactly like about this:

So far, supplying "-1" as parameter there lead to runtime error, which was quite reasonable contract. I have catched some bugs via that definition.

My experience is that you are quite often playing arithmetic games with "from", and sometimes they go wrong. Allowing negative values would make some of them silently hidden.

But I have to check the code for "GetCount()" in the "String::Find" before makeing any final decision.

BTW, if negative values are to be allowed with this semantics, we should probably extend this to other methods....

Mirek