Subject: substring find
Posted by hojtsy on Wed, 01 Feb 2006 14:10:34 GMT
View Forum Message <> Reply to Message

I can not seem to find any String::Find(String) method. Is it possible that this is missing? Then please add it to the wishlist.

Subject: Re: substring find
Posted by mirek on Wed, 01 Feb 2006 16:57:57 GMT
View Forum Message <> Reply to Message

Yes it is missing. Adding to ToDo... (there will huge String refactoring soon anyway...)

Mirek

Subject: Re: substring find
Posted by hojtsy on Thu, 02 Feb 2006 12:08:21 GMT
View Forum Message <> Reply to Message

What needs to be refactored in String? (Well except adding the substring and regexp find features, which doesn't seem to me as refactoring) Maybe I can help in it.

Subject: Re: substring find
Posted by mirek on Thu, 02 Feb 2006 19:28:40 GMT
View Forum Message <> Reply to Message

Implementation.

Right now, we have pure reference counted string. Not bad, but I now see better performing implementations that can save quite a lot of memory as well (small string optimization).

BTW, it would be interesting to combine substring search with character filter...

Mirek

Subject: Re: substring find
Posted by hojtsy on Mon, 06 Feb 2006 07:20:11 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 02 February 2006 14:28Right now, we have pure reference counted string. Not bad, but I now see better performing implementations that can save quite a lot of memory as well (small string optimization).

Is that small string optimization similar to what the Mitor class does? By the way where is the "Mitor" name coming from? I can't seem to find anything with this name on google.

---

Subject: Re: substring find
Posted by mirek on Mon, 06 Feb 2006 10:28:09 GMT
View Forum Message <> Reply to Message

No, small string optimization means that for small strings, you keep data inside String object, something like:

```
String {
  struct Large {
     const char *ptr;
     ....
  }
  union {
    char data[16]
    Large large_string;
  }
};
```

If you think about the issue, unshared reference counted string (and per my research, most reference counted strings are unshared) has 16 bytes overhead (4 bytes for pointer in String, 4 bytes reference count, 4 bytes length of string, 4 bytes allocation length). Then 70% of Strings has len < 15. Means, in 70% cases SSO will store the String "for free" when compared to current implementation. In remaining 30%, SSO will just use those 16 bytes to store pointer/length/alloc previously stored in shared string (storing there just reference count). Also, you will avoid alloc/free, interlocked increment/decrement etc...

Mirek

---

Subject: Re: substring find
Posted by mirek on Mon, 06 Feb 2006 10:30:20 GMT
View Forum Message <> Reply to Message

Mitor - Mi[niVec]tor

Not being native English speaker, I am sometimes too creative with words

The motivation: Most of Ctrls have just single Frame, but some have more than one. Speed of Mitor is less of concern there.

Mirek

---

Subject: Re: substring find
Posted by hojtsy on Mon, 06 Feb 2006 10:37:45 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 06 February 2006 05:28No, small string optimization means that for small strings, you keep data inside String object, something like:

```
String {
  struct Large {
     const char *ptr;
     ....
  }
  union {
    char data[16]
    Large large_string;
  }
};
```

Yes, that was what I thought. And it's similar to how the Mitor works - storing element data in the memory area of the container object itself.

Subject: Re: substring find
Posted by mirek on Mon, 06 Feb 2006 10:39:52 GMT
View Forum Message <> Reply to Message

Well, you are right. It was just that final performance characteristics are completely different, anyway, it is true that basic idea is the same.

Mirek