Subject: errors re strings
Posted by nlneilson on Fri, 29 Apr 2016 06:46:52 GMT
View Forum Message <> Reply to Message

(647): error C2664: 'std::basic_string<char,std::char_traits<char>,std::allocator
<char>>::basic_string(std::initializer_list<_Elem>,const std::allocator<ch
ar> &)': cannot convert argument 1 from 'Upp::String' to 'const
std::basic_string<char,std::char_traits<char>,std::allocator <char>> &'

A previous topic similar to this error was only when compiling with MINGW

This error is when recompiling an older app with upp 7901 and even with MSC15

Has the changes that causes this with C++11? or was this changes in the upp code?

Do I need to change all the 'string' to upp::String or is there an easier way ??
There are several apps that I may run into with this error as they are updated from a few years back.

here are a few lines of code:
```
 line 647   string Min (Tm, 2,2);
    string Sec (Tm, 4,2);
       String Tmz = Hr + ":" + Min + ":" + Sec;
    count<<=Tmz;
```

Neil

Subject: Re: errors re strings
Posted by mr_ped on Fri, 29 Apr 2016 08:44:17 GMT
View Forum Message <> Reply to Message

std::string and Upp::String are two completely different classes.

So the question is, why do you even use std::string in U++ application (or the other way around, why don't you stick solely to std::string, if you want to keep usage of U++ low and having that part of source usable without U++ Core too).

If your app is strongly U++ dependent, I would go for Upp::String only (except places where you call some external library, which has input as std::string, at that single point I would convert the String to string, but otherwise I would keep internally everything in String).

I often write clean C++ libraries in TheIDE (to be compiled on phone platforms, so I don't use Upp at all), then I use std::string only.

Your mixed way is certainly possible too, but I think it's not giving you any advantage, just

confusing.

Subject: Re: errors re strings
Posted by omari on Fri, 29 Apr 2016 08:59:05 GMT
View Forum Message <> Reply to Message

Hi,

the conversion from Upp::String to std::string is explicit by the function member ToStd()

```
String s_upp;
std::string s_std = s_upp.ToStd();
```

IMO the easier way is to avoid std::string, Upp::String is simpler, faster.

here a solution for this error:

```
String Min = Tm.Mid(2,2);
String Tmz = Hr + ":" + Min + ":" + Tm.Mid(4,2);
count<<=Tmz;
```

Subject: Re: errors re strings - SOLVED
Posted by nlneilson on Fri, 29 Apr 2016 11:24:41 GMT
View Forum Message <> Reply to Message

Hi mr ped thanks I did not know there was such a difference and that mixing them could lead to errors,
Most of my apps depend on several header .h 'apps' so I will try and remember what could cause problems.
I have no preference for which string or String, whatever works but I will try to use just one.

mr_ped wrote on Fri, 29 April 2016 01:44std::string and Upp::String are two completely different classes.

So the question is, why do you even use std::string in U++ application (or the other way around, why don't you stick solely to std::string, if you want to keep usage of U++ low and having that part of source usable without U++ Core too).

If your app is strongly U++ dependent, I would go for Upp::String only (except places where you call some external library, which has input as std::string, at that single point I would convert the String to string, but otherwise I would keep internally everything in String).

I often write clean C++ libraries in TheIDE (to be compiled on phone platforms, so I don't use Upp at all), then I use std::string only.

Your mixed way is certainly possible too, but I think it's not giving you any advantage, just confusing.

Neil

Subject: Re: errors re strings -
Posted by nlneilson on Fri, 29 Apr 2016 11:33:08 GMT
View Forum Message <> Reply to Message

omari wrote on Fri, 29 April 2016 01:59Hi,

the conversion from Upp::String to std::string is explicit by the function member ToStd()

```
String s_upp;
std::string s_std = s_upp.ToStd();
```

IMO the easier way is to avoid std::string, Upp::String is simpler, faster.

here a solution for this error:

```
String Min = Tm.Mid(2,2);
String Tmz = Hr + ":" + Min + ":" + Tm.Mid(4,2);
count<<=Tmz;
```

Hi Omari Thanks Your solution code worked first time and I appreciate the explanation, that is how I learn the most, code that works and then the reason behind it.

Neil

Subject: Re: errors re strings again or still
Posted by nlneilson on Wed, 25 May 2016 10:29:00 GMT
View Forum Message <> Reply to Message

I am still getting errors this time when compiling with MS15 or MINGW

Under upp 9701 it compiles OK with either compiler.

Under upp 9801 or 9861 it does not and I get
: error C2676: binary '<<=': 'Upp::String' does not define this operator or a conversion to a type acceptable to the predefined operator

ln<<= GGA2decAng(ln);

Subject: Re: errors re strings again or still
Posted by BioBytes on Wed, 25 May 2016 19:56:40 GMT
View Forum Message <> Reply to Message

Hi,

Quote:
I am still getting errors this time when compiling with MS15 or MINGW

Under upp 9701 it compiles OK with either compiler.
Under upp 9801 or 9861 it does not and I get
: error C2676: binary '<<=': 'Upp::String' does not define this operator or a conversion to a type acceptable to the predefined operator

ln<<= GGA2decAng(ln);

Normally
ln<<GGA2decAng(ln);
should solve your problem

Regards

Biobytes

Subject: Re: errors re strings again or still
Posted by nlneilson on Wed, 25 May 2016 21:45:39 GMT
View Forum Message <> Reply to Message

Thanks Biobytes

Just deleting the '=' character got rid of the error
