Subject: Using standard C++ in U++ application
Posted by eldiener on Fri, 06 Mar 2015 02:06:51 GMT
View Forum Message <> Reply to Message

Is there anything in U++ which will keep me from using standard C++ algorithms, containers, iterators and advanced libraries (Boost as an example) in an application where I am using U++ in order to create a cross-platform application using C++.

I realize that many U++ constructs use their own core libraries but I am talking outside of using those constructs with U++.

Subject: Re: Using standard C++ in U++ application
Posted by mirek on Fri, 06 Mar 2015 05:44:15 GMT
View Forum Message <> Reply to Message

No problem at all.

Usually you can mix it too, while U++ containers have different element requirements (either more constrained in Vector or totally relaxed in Array), there is huge intersection where std:: algorithms work on U++ containers, std:: containers work with U++ concrete types and U++ algorithms work on std:: containers.

Subject: Re: Using standard C++ in U++ application
Posted by eldiener on Fri, 06 Mar 2015 06:09:39 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 06 March 2015 00:44No problem at all.

Usually you can mix it too, while U++ containers have different element requirements (either more constrained in Vector or totally relaxed in Array), there is huge intersection where std:: algorithms work on U++ containers, std:: containers work with U++ concrete types and U++ algorithms work on std:: containers.

Thanks ! I will definitely take a look at U++ for cross-platform C++ programming.

Subject: Re: Using standard C++ in U++ application
Posted by mirek on Fri, 06 Mar 2015 07:48:40 GMT
View Forum Message <> Reply to Message

eldiener wrote on Fri, 06 March 2015 07:09mirek wrote on Fri, 06 March 2015 00:44No problem at all.

Usually you can mix it too, while U++ containers have different element requirements (either more constrained in Vector or totally relaxed in Array), there is huge intersection where std:: algorithms

work on U++ containers, std:: containers work with U++ concrete types and U++ algorithms work on std:: containers.

Thanks ! I will definitely take a look at U++ for cross-platform C++ programming.

PS.: The only thing prohibited is using std:: concretes in U++ Vector-flavor containers. E.g. Vector<std::string> is invalid, because std::string is not guaranteed to satisfy "moveability" constraint. This might cause some smaller issues, as you need to convert str::string to Upp::String before putting e.g. into U++ widgets.

OTOH, Upp::String has implicit conversion to/from std::string...
