Hi Mirek:

I understand that U++ currently aims to be compliant with c++14. Overall, U++ is very close to c++20 compliant.

IIRC, the only kind of complaints gcc/clang make when building TheIDE with std=c++20 is about caputuring `this` by default is deprecated in c++20 for a lambda. I am not sure if changing affected code to get rid of all such warnings will affect c++14 compliance but it's easy to make both worlds(or maybe all 3 worlds if we want to refer to c++17 and c++20 separately) happy anyways.

With MSVC, it is a different story. It complains in many cases like

    return some_condition ? SomeString : "Some ASCIIZ String";

These, though tedious, are easy to fix. I am no language lawyer, cannot tell which of MSVC and GCC/CLANG is/are correct here. But MSVC changes its behaviour from accepting it in C++14 to rejecting it in C++17 and beyond may tell something. Anyway it's not hard to make all worlds happy by just a little bit more keystrokes.

There are some more errors when compiling TheIDE with MSVC (mine is MSBT 2019 I believe) and std set to C++20. Another one is caused by Upp::Moveable::AssertMovealbe0 or something like that.

I mainly use CLANG now but I feel more assured if my code compiles fine on Ubuntu with GCC & CLANG, and on Windows with MS c++ compiler. I don't know if other users think this kind of check has some value, but it likely will be welcomed if a user can use u++ with more recent standard if he/she wishes(so that he/she can embrace utilities like constexpr-if and concept), and with the compiler he/she choose (one of the 3 major), while the bulk of U++ is in c++14 and be backward compatible.

Correction and some detailed error message:
1. The MSVC I used is MSBT22x64
2. The error message with AssertMoveable0 is like
Quote:
C:\upp\uppsrc\Core\Topt.h (157): error C2100: illegal indirection
C:\upp\uppsrc\Core\Topt.h (172): note: see reference to function template instantiation 'void Upp::AssertMoveable0<T>(T *)' being compiled
    with
    [
    T=double
    ]ChWin32.cpp

BR,
Lance