Subject: Tricky template non-typeParameter
Posted by Xemuth on Sat, 08 Jun 2019 14:36:55 GMT
View Forum Message <> Reply to Message

Hello,

Is it possible to do something like that :

```
using namespace Upp;

template<dword> struct Type_selector;
template<> struct Type_selector<1> { using Type = int;};
template<> struct Type_selector<3> { using Type = String;};
template<> struct Type_selector<4> { using Type = Date;};
template<> struct Type_selector<134217728> { using Type = long;};
template<> struct Type_selector<134217729> { using Type = float;};
template<> struct Type_selector<2> { using Type = double;};
template<> struct Type_selector<11> { using Type = bool;};

template<dword type> using Type = typename Type_selector<type>::Type;

CONSOLE_APP_MAIN
{
 Type_selector<1>::Type E =1; //work
 Type<1> T =2;//Work

 // Is it possible to actually doing something like that :
 Type<"int"> F = 3; //I want F to be Int  from giving Upp::String to 'Type'
}
```

thanks in advance

best regard

Subject: Re: Tricky template non-typeParameter
Posted by Novo on Sat, 08 Jun 2019 15:27:45 GMT
View Forum Message <> Reply to Message

According to this Non-type template parameter can be

- std::nullptr_t (since C++11);
- an integral type;
- a pointer type (to object or to function);
- a pointer to member type (to member object or to member function);
- an enumeration type.

It cannot be string (const char*) or double/float.
Try to use enums instead ...
