Subject: Value with type float
Posted by abductee23 on Thu, 03 Mar 2022 12:34:03 GMT
View Forum Message <> Reply to Message

Hi,

i was wondering - when will float be supported by Value?
(couldn't find it in the roadmap)

in the interrim - is there a bazar or patch thingy one could point me to?

thanks in advance

--m

Subject: Re: Value with type float
Posted by mirek on Sat, 19 Mar 2022 09:38:02 GMT
View Forum Message <> Reply to Message

abductee23 wrote on Thu, 03 March 2022 13:34Hi,

i was wondering - when will float be supported by Value?
(couldn't find it in the roadmap)

in the interrim - is there a bazar or patch thingy one could point me to?

thanks in advance

--m

Supporting float directly does not make much sense as you can store float to double and restore back without loosing any precision.

If you are about memory consumption, both would consume the same (sizeof(Value) is 16 and double is stored directly into 8 of those 16 bytes).

Now counterexample can be int vs int64. However, we still support 32-bit CPUs and OSes where int64 was HW type. double is HW on all platforms we dare to support...

Mirek

Subject: Re: Value with type float
Posted by abductee23 on Thu, 31 Mar 2022 09:51:34 GMT
View Forum Message <> Reply to Message

Hi Mirek,

you have no idea how much it pains me to read that.
i was very excited when i found upp back in the day 2005 ish - and realized that i have to never worry about maintaining my own ux library... and can focus on what i do best: graphics programming.

maybe i am seeing this too much from my own viewpoint - graphics?

before upp i viewed c++ as a "necessary evil" - you have with your code, and esp your forum posts have quiet an impact.
and you seem to be so many times on what i consider "the side of an argument" when it comes to technical things.

and - to be blunt - only in recent times the c++ world has caught up to where upp was a decade ago (somehow they celebrate it by doing to their standart whatever that is they are doing...)

ok, just to make sure i don't get misinterpreted as a "hey i can't start using your stuff until you give me X Y and Z"-person(you probably have had a lot of these)

anyways - over the last 15-ish years there was not a project where not on some level i either quickly hacked together something in upp - or made it a upp thing to begin with - that goes for work and private stuff.(see pictures below - only private projects stuff obviously)

beeing able to crank out a tool in a day or less has been an invaluable asset to my work and private coding life.
(the "just drag all the pngs to this tool and if will fix the alpha channel"-incident comes to mind ^^)

and the only actual problem that i am facing over and over again: floats - love'em or hate'em - the reality is they are essential to computergraphics.

dealing with float issiues/precission is a time honered "tradition" for cg and audio(esp snyth) programmers :)

having a flot in memory - writing it out through value/json , then reading it again incurs a conversion and its not the same number - (i guess on a personal level i am probably puzzled how this is not bothering you - but thats beside the point)

of course - i do not disagree that that more precision is better and preferred in a lot of cases - but gpus have a significant performance penalty for using doubles and there's nothing i can do about that.(opengl/d3d are also very creative when it comes to IEEE754)

so i guess my questions are: do you see any wiggleroom when it comes to Value/Jsonize?

I want to make sure its understood this is coming from a a good place and is a genuine request - and i guess say thank you for the last 15 yrs of being there and improving upp.

File Attachments
1) amboss.png, downloaded 168 times
2) nvscene.png, downloaded 155 times
3) the_dude_abides.png, downloaded 151 times

Subject: Re: Value with type float
Posted by Tom1 on Thu, 31 Mar 2022 11:49:37 GMT
View Forum Message <> Reply to Message

Hi,

I could see some benefit for supporting float in Value too. (I do signal processing with floats, so they frequently end up in various locations in my code.)
Having Value supporting float and EditFloat/EditFloatSpin added would allow cleaner code with dialogs.

Currently I have to round the float value to a clean double with roundr() to avoid excessive decimal places in EditDouble/EditDoubleSpin display. Also, when reading the value out from EditDouble/EditDoubleSpin, I will need to cast first to (double) and only thereafter to (float).

As an example, I have filtering frequencies controlled with:
void SetHPF(float fc);
float GetHPF();
Filling the EditDoubleSpin:
hpf<<=roundr(GetHPF(),3);
Reading the EditDoubleSpin:
hpf.WhenAction=[&](){ SetHPF((float)(double)~hpf); };

With float Value and EditFloatSpin support I would expect to work with:
hpf<<=GetHPF();
And:
hpf.WhenAction=[&](){ SetHPF((float)~hpf); };
Anyway, changes in Core/CtrlCore/CtrlLib are something for Mirek to decide.

Best regards,

Tom

Subject: Re: Value with type float
Posted by Tom1 on Thu, 31 Mar 2022 13:14:15 GMT
View Forum Message <> Reply to Message

Hi,

Now here's an uneducated, quick and dirty attempt to support float as Value.

DISCLAIMER: This will likely break something and cause all kinds of trouble, as I do not understand most of the fine details of Value/XML/JSON or Core for that matter. So, if you wish to take the risk and test this, please remember to revert back to official sources before getting back to serious work. And also destroy your copy of these attached files.

Best regards,

Tom

File Attachments
1) FloatValuePrototype.zip, downloaded 132 times

Subject: Re: Value with type float
Posted by mirek on Fri, 01 Apr 2022 09:03:29 GMT
View Forum Message <> Reply to Message

abductee23 wrote on Thu, 31 March 2022 11:51
having a flot in memory - writing it out through value/json , then reading it again incurs a conversion and its not the same number - (i guess on a personal level i am probably puzzled how this is not bothering you - but thats beside the point)

But that is altogether different issue! Converting float or double to decimal text and back involves completely different set of problems.

Frankly, current "stable" version (2021.1) has this botched up a bit. Here the IEEE standard states that 15 digit double numbers must be converted from text and back without a change. We were short on this promise, but this is hopefully fixed in the upcoming version.

Anyway, conversion from float to double is basically only adding zeroes to mantissa, converting from double back to float involves in most cases (*) just rounding the same additional bits (if they are zero, rounding is obviously down).

(*) - there is of course issue of range (float is about e-38 to e+38 while double e-308 to e-308) and denormals, but that is in most cases bedides the point..

Mirek

Subject: Re: Value with type float
Posted by mirek on Fri, 01 Apr 2022 09:07:33 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 31 March 2022 13:49Hi,

I could see some benefit for supporting float in Value too. (I do signal processing with floats, so they frequently end up in various locations in my code.)
Having Value supporting float and EditFloat/EditFloatSpin added would allow cleaner code with dialogs.

Currently I have to round the float value to a clean double with roundr() to avoid excessive decimal places in EditDouble/EditDoubleSpin display. Also, when reading the value out from EditDouble/EditDoubleSpin, I will need to cast first to (double) and only thereafter to (float).

As an example, I have filtering frequencies controlled with:
void SetHPF(float fc);
float GetHPF();
Filling the EditDoubleSpin:
hpf<<=roundr(GetHPF(),3);

Uhm, roundr is sort of sin of past. Perhaps it is a bad idea to pretend that either float or double are "decimal"...

That said, maybe we should just add operator float to Value and constructor from float (if that one is even needed)?

That would solve most of those additional casts that you do not like.

Mirek

Subject: Re: Value with type float
Posted by Tom1 on Fri, 01 Apr 2022 09:13:58 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Would that solve my EditDoubleSpin showing 0.002000000095 instead of 0.002 when I remove the roundr(xxx,3)?

Best regards,

Tom

Subject: Re: Value with type float
Posted by Tom1 on Fri, 01 Apr 2022 10:06:35 GMT
View Forum Message <> Reply to Message

I see. Simply adding operator float to Value:
```
 operator float() const          { return (float)(double)*this; }
```
removes need for any casting at all. The required code gets clean:
```
hpf.WhenAction=[&](){ SetHPF(~hpf); };
```
EDIT: Removed from here my stupid idea to round float when constructing a Value.

Best regards,

Tom

Subject: Re: Value with type float
Posted by Tom1 on Fri, 01 Apr 2022 12:03:16 GMT
View Forum Message <> Reply to Message

Hi,

The following code portrays the float rounding issue:
```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 Value v(0.002f);
 Cout() << v << "\n";
 Cout() << FormatDouble(v) << "\n";
 Cout() << FormatG(v,7) << "\n";
}
```

The result of running the above follows:
```
0.0020000000949949
0.0020000000949949
0.002
<--- Finished, press [ENTER] to close the window --->
```

I do not know how to solve this cleanly. In any case a regular user seeing 0.002000000095 in a

field where he expects to see 0.002, will not be happy about it. For years I have used roundr() all over the code to fix this up, but having a solution hidden in Core would be awesome! :)

Best regards,

Tom

Subject: Re: Value with type float
Posted by Tom1 on Fri, 01 Apr 2022 13:26:36 GMT
View Forum Message <> Reply to Message

OK, here's an idea to handle the EditDouble/EditDoubleSpin rounding issue. Adding something like:
hpf.Pattern("%.7g");
to the constructor of the dialog. This will yield possibly sufficient rounding to the float value being represented as double to avoid excessive decimals.

Of course it would be nice to have such 'float compatibility' Pattern -feature available as a flag / checkbox in layout editor for EditDouble, EditDoubleSpin and the NotNull relatives.

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Fri, 01 Apr 2022 23:31:06 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 01 April 2022 14:03Hi,

The following code portrays the float rounding issue:

```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 Value v(0.002f);
 Cout() << v << "\n";
 Cout() << FormatDouble(v) << "\n";
 Cout() << FormatG(v,7) << "\n";
}
```

The result of running the above follows:

0.0020000000949949
0.0020000000949949
0.002
<--- Finished, press [ENTER] to close the window --->

I do not know how to solve this cleanly. In any case a regular user seeing 0.002000000095 in a field where he expects to see 0.002, will not be happy about it. For years I have used roundr() all over the code to fix this up, but having a solution hidden in Core would be awesome! :)

Best regards,

Tom

This can actually be a bug in formatting routine. I will check ASAP.

Mirek

Subject: Re: Value with type float
Posted by mirek on Fri, 01 Apr 2022 23:31:58 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 01 April 2022 15:26OK, here's an idea to handle the EditDouble/EditDoubleSpin rounding issue. Adding something like:
hpf.Pattern("%.7g");
to the constructor of the dialog. This will yield possibly sufficient rounding to the float value being represented as double to avoid excessive decimals.

Yes, this is the correct solution - do decimal rounding when the number is actually converted to decimals.

Mirek

Subject: Re: Value with type float
Posted by mirek on Sat, 02 Apr 2022 07:43:44 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 01 April 2022 14:03Hi,

The following code portrays the float rounding issue:
#include <Core/Core.h>

using namespace Upp;

```
CONSOLE_APP_MAIN
{
 Value v(0.002f);
 Cout() << v << "\n";
 Cout() << FormatDouble(v) << "\n";
 Cout() << FormatG(v,7) << "\n";
}
```

The result of running the above follows:
```
0.0020000000949949
0.0020000000949949
0.002
<--- Finished, press [ENTER] to close the window --->
```

I do not know how to solve this cleanly. In any case a regular user seeing 0.002000000095 in a field where he expects to see 0.002, will not be happy about it. For years I have used roundr() all over the code to fix this up, but having a solution hidden in Core would be awesome! :)

Best regards,

Tom

OK, so it is correct behaviour. You can simplify that to

```
 double x = 0.002;
 DDUMP(x);
 float fx = x;
 x = fx;
 DDUMP(x);
```

which produces exactly the same result. Now for an explanation what is going on here:

0.002 cannot be exactly represented as double. Normal double formatting (as used in DDUMP) rounds for 15 decimal digits which is guaranteed precision and it all yields a "correct" result (in both directions, closes value is choosen and everything is "fine").

Anyway, when you convert it to float, you have to round at equivalent of about 7 valid digits (you have to cut 28 bits of mantissa). In this case, the last mantissa bit of float is rounded up to 1 as that is the closest value to original double. When converting back to double, this cannot be undone, hence you get those "949949" digits at the end.

That said, all of this made me think that if we knew that it is float in Value, we might be able to apply for something like FormatG(v,7) instead when displaying it (e.g. in EditDouble).

Mirek

Subject: Re: Value with type float
Posted by Tom1 on Sat, 02 Apr 2022 08:55:05 GMT
View Forum Message <> Reply to Message

Hi Mirek,
Quote:That said, all of this made me think that if we knew that it is float in Value, we might be able to apply for something like FormatG(v,7) instead when displaying it (e.g. in EditDouble).
Would that involve introducing:
const dword FLOAT_V = 13;
and 'DOUBLE_V' -like processing for it, except when putting it out in FormatG() and friends?

This is in some ways pretty much what I attempted to do with the above "float Value prototype" above, but that is really bad as I do not really understand all the details of Value. You may wish to take a look anyway -- and then trash it. :)

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Sat, 02 Apr 2022 09:26:24 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 02 April 2022 10:55Hi Mirek,
Quote:That said, all of this made me think that if we knew that it is float in Value, we might be able to apply for something like FormatG(v,7) instead when displaying it (e.g. in EditDouble).
Would that involve introducing:
const dword FLOAT_V = 13;
and 'DOUBLE_V' -like processing for it, except when putting it out in FormatG() and friends?

This is in some ways pretty much what I attempted to do with the above "float Value prototype" above, but that is really bad as I do not really understand all the details of Value. You may wish to take a look anyway -- and then trash it. :)

Best regards,

Tom

Well, I guess I have to sleep over this.... FormatG(v, 7) being the only difference in processing there seems a bit too little...

Mirek

Subject: Re: Value with type float
Posted by Tom1 on Sat, 02 Apr 2022 11:31:38 GMT

View Forum Message <> Reply to Message

No problem... I trust you will soon come up with some astonishingly smart three line solution for the task! :)

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Mon, 11 Apr 2022 14:52:07 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 02 April 2022 13:31No problem... I trust you will soon come up with some astonishingly smart three line solution for the task! :)

Best regards,

Tom

Nothing too smart for now, but I have added ConvertFloat and EditFloat.

The jury is still out for float in Value.... do not want to do that now. It feels like while it fixes some issues (e.g. Value->JSON conversion when the values went through float precision), there might be many caveats.

Subject: Re: Value with type float
Posted by Tom1 on Tue, 12 Apr 2022 07:26:00 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 11 April 2022 17:52Tom1 wrote on Sat, 02 April 2022 13:31No problem... I trust you will soon come up with some astonishingly smart three line solution for the task! :)

Best regards,

Tom

Nothing too smart for now, but I have added ConvertFloat and EditFloat.

The jury is still out for float in Value.... do not want to do that now. It feels like while it fixes some issues (e.g. Value->JSON conversion when the values went through float precision), there might be many caveats.

Hi Mirek,

Thanks! EditFloat works fine now. However, there are a couple of things more. First, could you

add float operator to Value for easy reading of EditFloat:
operator float() const { return Is(DOUBLE_V) ? (float)GetSmallRaw<double>() : (float)GetOtherDouble(); }
Second, can you add EditFloatSpin and EditFloatNotNullSpin variants?:
typedef WithSpin<float, EditFloat> EditFloatSpin;
typedef WithSpin<float, EditFloatNotNull> EditFloatNotNullSpin;

// And then some magic to make those two working and visible in layout editor.
Thanks and best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Tue, 12 Apr 2022 11:40:09 GMT
View Forum Message <> Reply to Message

Done.

Subject: Re: Value with type float
Posted by Tom1 on Tue, 12 Apr 2022 11:54:39 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks! This works great!

Best regards,

Tom

Subject: Re: Value with type float
Posted by Tom1 on Tue, 10 May 2022 11:21:06 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Sorry to return to this subject, but could you consider adding Null support for float?

I had something like this in mind... (in Core/Defs.h):
const int INT_NULL = INT_MIN;
const int64 INT64_NULL = INT64_MIN;

constexpr double DOUBLE_NULL = -std::numeric_limits<double>::infinity();
constexpr float FLOAT_NULL = -std::numeric_limits<float>::infinity();

```
class Nuller {
public:
 operator int() const              { return INT_NULL; }
 operator int64() const             { return INT64_NULL; }
 operator double() const              { return DOUBLE_NULL; }
 operator float() const             { return FLOAT_NULL; }
 operator bool() const              { return false; }

 Nuller() {}
};

extern const Nuller Null;

template <class T> void SetNull(T& x) { x = Null; }

template <class T> bool IsNull(const T& x)       { return x.IsNullInstance(); }

template<> inline bool  IsNull(const int& i)     { return i == INT_NULL; }
template<> inline bool  IsNull(const int64& i)   { return i == INT64_NULL; }
template<> inline bool  IsNull(const double& r)  { return !(std::abs(r) <
std::numeric_limits<double>::infinity()); }
template<> inline bool  IsNull(const float& r)   { return !(std::abs(r) <
std::numeric_limits<float>::infinity()); }
template<> inline bool  IsNull(const bool& r  )  { return false; }
```

Although, I'm not entirely sure, if this is completely correct way to do it.

Best regards,

Tom

PS. EDIT: I think the above should work mostly. Only the "Cout() << FLOAT_NULL;" prints -inf instead of empty field, which is the default for "Cout() << DOUBLE_NULL;":

```
CONSOLE_APP_MAIN{
 double d=Null;
 float f=Null;
 double a=f;
 float b=d;
 Cout() << "d = " << d << "\n";
 Cout() << "f = " << f << "\n";
 Cout() << "a = " << a << "\n";
 Cout() << "b = " << b << "\n";
 Cout() << "a == f : " << (bool)(a==f) << "\n";
 Cout() << "b == d : " << (bool)(b==d) << "\n";
 Cout() << "IsNull(d) = " << IsNull(d) << "\n";
 Cout() << "IsNull(f) = " << IsNull(f) << "\n";
```

```
 Cout() << "IsNull(a) = " << IsNull(a) << "\n";
 Cout() << "IsNull(b) = " << IsNull(b) << "\n";
}
```

Subject: Re: Value with type float
Posted by mirek on Tue, 10 May 2022 13:44:16 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 10 May 2022 13:21Hi Mirek,

Sorry to return to this subject, but could you consider adding Null support for float?

I had something like this in mind... (in Core/Defs.h):

```
const int    INT_NULL         =    INT_MIN;
const int64  INT64_NULL       =    INT64_MIN;

constexpr double DOUBLE_NULL    =    -std::numeric_limits<double>::infinity();
constexpr float FLOAT_NULL      =    -std::numeric_limits<float>::infinity();

class Nuller {
public:
 operator int() const              { return INT_NULL; }
 operator int64() const             { return INT64_NULL; }
 operator double() const             { return DOUBLE_NULL; }
 operator float() const             { return FLOAT_NULL; }
 operator bool() const              { return false; }

 Nuller() {}
};

extern const Nuller Null;

template <class T> void SetNull(T& x) { x = Null; }

template <class T> bool IsNull(const T& x)       { return x.IsNullInstance(); }

template<> inline bool  IsNull(const int& i)     { return i == INT_NULL; }
template<> inline bool  IsNull(const int64& i)   { return i == INT64_NULL; }
template<> inline bool  IsNull(const double& r)  { return !(std::abs(r) <
std::numeric_limits<double>::infinity()); }
template<> inline bool  IsNull(const float& r)   { return !(std::abs(r) <
std::numeric_limits<float>::infinity()); }
template<> inline bool  IsNull(const bool& r  )  { return false; }
```

Although, I'm not entirely sure, if this is completely correct way to do it.

Best regards,

Tom

PS. EDIT: I think the above should work mostly. Only the "Cout() << FLOAT_NULL;" prints -inf instead of empty field, which is the default for "Cout() << DOUBLE_NULL;":

```
CONSOLE_APP_MAIN{
 double d=Null;
 float f=Null;
 double a=f;
 float b=d;
 Cout() << "d = " << d << "\n";
 Cout() << "f = " << f << "\n";
 Cout() << "a = " << a << "\n";
 Cout() << "b = " << b << "\n";
 Cout() << "a == f : " << (bool)(a==f) << "\n";
 Cout() << "b == d : " << (bool)(b==d) << "\n";
 Cout() << "IsNull(d) = " << IsNull(d) << "\n";
 Cout() << "IsNull(f) = " << IsNull(f) << "\n";
 Cout() << "IsNull(a) = " << IsNull(a) << "\n";
 Cout() << "IsNull(b) = " << IsNull(b) << "\n";
}
```

IDK. Does it solve any real problem?

Up until this year, I have considered double-float relation to be similar to int-int16. You use float to reduce memory consumption or for special things (GPU), but you really do not need to support it in Value or widgets. Served me well for many many years. What has changed? :)

Mirek

Subject: Re: Value with type float
Posted by Tom1 on Tue, 10 May 2022 18:08:46 GMT
View Forum Message <> Reply to Message

Hi,

What has changed is that I have really started to enjoy the new EditFloatSpin! :)

As a result, initializing and reading my float variables to/from EditFloatSpin (as used in e.g. filter parameters) is clean and easy. However, I found that initializing my EditFloatSpin to empty field (e.g. when such filtering is not currently used) requires my float variable to be at Null:

```
EditFloatSpin hpfedit;
...
float hpf=(float)(double)Null;
```

...
hpfedit <<= hpf;
Alternatively, some value of hpf (e.g.< 0) could be interpreted as empty filtering and I could do hpfedit.Clear(); if such is the case.

However, I prefer nice, short and clean code, so I would like to write just:
float hpf=Null;
I.e. without any type cast. So, in effect, this is just to make the code cleaner.

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Tue, 10 May 2022 19:26:38 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 10 May 2022 20:08Hi,

What has changed is that I have really started to enjoy the new EditFloatSpin! :)

As a result, initializing and reading my float variables to/from EditFloatSpin (as used in e.g. filter parameters) is clean and easy. However, I found that initializing my EditFloatSpin to empty field (e.g. when such filtering is not currently used) requires my float variable to be at Null:
EditFloatSpin hpfedit;
...
float hpf=(float)(double)Null;
...
hpfedit <<= hpf;
Alternatively, some value of hpf (e.g.< 0) could be interpreted as empty filtering and I could do hpfedit.Clear(); if such is the case.

However, I prefer nice, short and clean code, so I would like to write just:
float hpf=Null;
I.e. without any type cast. So, in effect, this is just to make the code cleaner.

Best regards,

Tom

Why dont you just use

double hpf;

?

Subject: Re: Value with type float
Posted by Tom1 on Wed, 11 May 2022 06:20:56 GMT
View Forum Message <> Reply to Message

Quote:
Why dont you just use

double hpf;

?
Well, all my signal processing code runs on 32-bit floats, and therefore, the various coefficients/parameters are also floats. It is straight forward to keep it up in the user interface too. If I used doubles in the GUI, I would end up converting forth and back all those parameters. Never wish to go back there, now that I have EditFloat and EditFloatSpin! :)

Anyway, if you feel seriously reluctant to add Null support for float, I can live with it: I figured out a way to do it outside of Core almost cleanly:
constexpr float Nullf = -std::numeric_limits<float>::infinity();
inline bool IsNull(const float& r) { return !(std::abs(r) < std::numeric_limits<float>::infinity()); }
inline void SetNull(float& x) { x = Nullf; }
Still, it would be nicer inside Core... After all, it would introduce only three lines of new code in Core/Defs.h.

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Fri, 03 Jun 2022 08:52:52 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 11 May 2022 08:20Quote:
Why dont you just use

double hpf;

?
Well, all my signal processing code runs on 32-bit floats, and therefore, the various coefficients/parameters are also floats. It is straight forward to keep it up in the user interface too. If I used doubles in the GUI, I would end up converting forth and back all those parameters. Never wish to go back there, now that I have EditFloat and EditFloatSpin! :)

Anyway, if you feel seriously reluctant to add Null support for float, I can live with it: I figured out a way to do it outside of Core almost cleanly:
constexpr float Nullf = -std::numeric_limits<float>::infinity();
inline bool IsNull(const float& r) { return !(std::abs(r) < std::numeric_limits<float>::infinity()); }
inline void SetNull(float& x) { x = Nullf; }

Still, it would be nicer inside Core... After all, it would introduce only three lines of new code in Core/Defs.h.

Best regards,

Tom

Got a new idea for a bit more universal solution:

The most important difference between Value(double) and Value(float) is the precision used when formatting it. What about to add a general precision hint to Value(double)? That would make possible to e.g. specify arbitrary precision for Json output etc...

Subject: Re: Value with type float
Posted by Tom1 on Fri, 03 Jun 2022 10:03:32 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Sounds interesting, but I cannot really figure out all consequences this change may have. If I may express my needs from the user's viewpoint, I hope:

- I can use Value with float as easily as with double without explicit type casts anywhere
- Assigning a float to Value reads back from Value exactly the same as it went in
- If EditFloat* is again replaced by EditDouble*, the behavior is exactly the same as it is now with EditFloat*
- float supports Null

Maybe the precision hint you are suggesting could be automatically initialized in Value(double) / Value(float) constructors and assignment operators =(double) / =(float) to suit the assigned data type?

Best regards,

Tom

Subject: Re: Value with type float
Posted by jimlef on Mon, 05 Sep 2022 23:41:02 GMT
View Forum Message <> Reply to Message

Related to all this, the innacuracies due to the way the pc handles float/double numbers have been ... irking me as well. I was thinking of float instead of double, but that doesn't fix the issue (and there isn't any json support for float anyway). In my case, I just want sales tax to be 0.08 (or whatever, it's stored in json) - not 0.080000000000000002. I've tried to utilize FormatG and round functions to no joy. My solution has been to store and use ints instead of floating point, and do

conversions in the program where needed. It handles positive and negative values effectively enough for my needs, and my little app was trivial to rework. Anyone wishing to torture themselves with my coding can view it at https://github.com/phoenixjim/Invoices :lol: . Some relevant portions are in config.cpp/h and the report files (In the reports folder) regarding how this is used. Convert.h/cpp has a few altered convert functions to match...

It's great how supportive the folks on this forum are - I hope this helps someone else :)

Subject: Re: Value with type float
Posted by Oblivion on Tue, 06 Sep 2022 20:42:06 GMT
View Forum Message <> Reply to Message

Hello jimlef,

Do you mean this?

```
RLOG(Format(
 "Pi(2): %.02f\n"
 "PI(4): %.04f\n"
 "PI(8): %.08f\n",
 roundr(M_PI, 2),
 roundr(M_PI, 4),
 roundr(M_PI, 8)
 ));

double tax = 0.0800000000123;
String stax = FormatG(tax, 3);
RLOG("Formatted tax: " << stax);
RLOG("Scanned   tax: " << ScanDouble(stax));

Json jtax_r("tax-rounded", roundr(tax, 2));
Json jtax_n("tax", tax);

RDUMP(jtax_r);
RDUMP(jtax_n);
```

```
Pi(2): 3.10
PI(4): 3.1420
PI(8): 3.14159270

Formatted tax: 0.08
Scanned   tax: 0.08
jtax_r = {"tax-rounded":0.08}
```

jtax_n = {"tax":0.0800000000123}

[Edit: Code updated.]

Best regards,
Oblivion

Subject: Re: Value with type float
Posted by jimlef on Thu, 08 Sep 2022 12:49:19 GMT
View Forum Message <> Reply to Message

Yes 8) That would be it I'm thinking

Thank you!

Jim

Oblivion wrote on Tue, 06 September 2022 16:42Hello jimlef,

Do you mean this?

```
RLOG(Format(
 "Pi(2): %.02f\n"
 "PI(4): %.04f\n"
 "PI(8): %.08f\n",
 roundr(M_PI, 2),
 roundr(M_PI, 4),
 roundr(M_PI, 8)
 ));

double tax = 0.0800000000123;
String stax = FormatG(tax, 3);
RLOG("Formatted tax: " << stax);
RLOG("Scanned   tax: " << ScanDouble(stax));

Json jtax_r("tax-rounded", roundr(tax, 2));
Json jtax_n("tax", tax);

RDUMP(jtax_r);
RDUMP(jtax_n);
```

Pi(2): 3.10

PI(4): 3.1420
PI(8): 3.14159270

Formatted tax: 0.08
Scanned tax: 0.08
jtax_r = {"tax-rounded":0.08}
jtax_n = {"tax":0.0800000000123}

[Edit: Code updated.]

Best regards,
Oblivion

Subject: Re: Value with type float
Posted by mirek on Wed, 04 Oct 2023 07:55:20 GMT
View Forum Message <> Reply to Message

Just revisting this issue...

Tom1 wrote on Fri, 03 June 2022 12:03
- I can use Value with float as easily as with double without explicit type casts anywhere

Yes.

Quote:
- Assigning a float to Value reads back from Value exactly the same as it went in

If "reads back" means text output, then yes.

Quote:
- If EditFloat* is again replaced by EditDouble*, the behavior is exactly the same as it is now with EditFloat*

Should work.

Quote:
- float supports Null

No. Is that a problem?

Quote:

Maybe the precision hint you are suggesting could be automatically initialized in Value(double) / Value(float) constructors and assignment operators =(double) / =(float) to suit the assigned data type?

That was exactly the plan.

I think advantage of this solution is that it allows you to specify arbitrary precision for any double-holding Value. E.g.

```
double x = 3.14159;
Value json;
json("pi") = Precision(x, 2); // Precision does not exist yet
DDUMP(AsJSON(json));
```

{"pi":3.14}

Still an idea though.

Subject: Re: Value with type float
Posted by Tom1 on Wed, 04 Oct 2023 10:08:40 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 04 October 2023 10:55Just revisting this issue...

Tom1 wrote on Fri, 03 June 2022 12:03
- I can use Value with float as easily as with double without explicit type casts anywhere

Yes.

Quote:
- Assigning a float to Value reads back from Value exactly the same as it went in

If "reads back" means text output, then yes.

Quote:
- If EditFloat* is again replaced by EditDouble*, the behavior is exactly the same as it is now with EditFloat*

Should work.

Quote:
- float supports Null

No. Is that a problem?

Quote:
Maybe the precision hint you are suggesting could be automatically initialized in Value(double) / Value(float) constructors and assignment operators =(double) / =(float) to suit the assigned data type?

That was exactly the plan.

I think advantage of this solution is that it allows you to specify arbitrary precision for any double-holding Value. E.g.

```
double x = 3.14159;
Value json;
json("pi") = Precision(x, 2); // Precision does not exist yet
DDUMP(AsJSON(json));
```

{"pi":3.14}

Still an idea though.

Hi Mirek,

Thanks for the update on this subject.

For me 'reads back exactly the same' means binary equality:

```
float a = 1.234562f;
float b = 1.234562f;
Value c = b;
b = c;
if(a == b) Cout() << "Great, it works!\r\n"
else Cout() << "No, it does not read back the same...\r\n";
```

As for float Null, yes, please include the changes shown in:

https://www.ultimatepp.org/forums/index.php?t=msg&th=120 82&goto=59874&#msg_59874

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Fri, 06 Oct 2023 10:51:18 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 04 October 2023 12:08mirek wrote on Wed, 04 October 2023 10:55Just revisting this issue...

Tom1 wrote on Fri, 03 June 2022 12:03
- I can use Value with float as easily as with double without explicit type casts anywhere

Yes.

Quote:
- Assigning a float to Value reads back from Value exactly the same as it went in

If "reads back" means text output, then yes.

Quote:
- If EditFloat* is again replaced by EditDouble*, the behavior is exactly the same as it is now with EditFloat*

Should work.

Quote:
- float supports Null

No. Is that a problem?

Quote:
Maybe the precision hint you are suggesting could be automatically initialized in Value(double) / Value(float) constructors and assignment operators =(double) / =(float) to suit the assigned data type?

That was exactly the plan.

I think advantage of this solution is that it allows you to specify arbitrary precision for any double-holding Value. E.g.

```
double x = 3.14159;
Value json;
json("pi") = Precision(x, 2); // Precision does not exist yet
DDUMP(AsJSON(json));
```

{"pi":3.14}

Still an idea though.

Hi Mirek,

Thanks for the update on this subject.

For me 'reads back exactly the same' means binary equality:

```
float a = 1.234562f;
float b = 1.234562f;
Value c = b;
b = c;
if(a == b) Cout() << "Great, it works!\r\n"
else Cout() << "No, it does not read back the same...\r\n";
```

This works (and I believe it should) right now. Am I missing something?

Quote:

As for float Null, yes, please include the changes shown in:

https://www.ultimatepp.org/forums/index.php?t=msg&th=120 82&goto=59874&#msg_59874

I am reluctant adding yet another Null...

Mirek

Subject: Re: Value with type float
Posted by Tom1 on Fri, 06 Oct 2023 11:40:07 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 06 October 2023 13:51This works (and I believe it should) right now. Am I

missing something?

Quote:

As for float Null, yes, please include the changes shown in:

https://www.ultimatepp.org/forums/index.php?t=msg&th=120 82&goto=59874&#msg_59874

I am reluctant adding yet another Null...

Mirek

Hi Mirek,

No, you're not missing anything here. My code was there just to demonstrate what I mean by 'reads back exactly the same'.

As for the float Null: Instead of writing this:

```
float f = (float)(double)Null;
...
Cout() << IsNull((double)f) << "\r\n";
```

I would like to do this:

```
float f = Null;
...
Cout() << IsNull(f) << "\r\n";
```

Is there a specific reason why you wish to avoid adding another Null to support float?

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Fri, 06 Oct 2023 11:54:54 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 06 October 2023 13:40mirek wrote on Fri, 06 October 2023 13:51This works (and I believe it should) right now. Am I missing something?

Quote:

As for float Null, yes, please include the changes shown in:

I am reluctant adding yet another Null...

Mirek

Hi Mirek,

No, you're not missing anything here. My code was there just to demonstrate what I mean by 'reads back exactly the same'.

I am asking because it looks like you had problems with this in the past....

Quote:
Is there a specific reason why you wish to avoid adding another Null to support float?

Reluctance to add features that are not necessarry nor very helpful (not that U++ is not already full of them, but still).

I still see no reason to use float instead of double except for saving the storage space...

Mirek

Subject: Re: Value with type float
Posted by Tom1 on Fri, 06 Oct 2023 12:28:13 GMT
View Forum Message <> Reply to Message

Yes, for storage space, RAM space, disk bandwidth (for speed), memory bandwidth (for speed), CPU cache space (for speed), ... I use floats all the time for any large data sets where float's accuracy is sufficient. It really pays off. Also, I mark missing data/observations using '(float)(double)Null' to handle the situation properly. And thanks to your efforts float edit fields, these days it is also handy to pull the float observations directly to dialogs for editing.

I think float is and will still remain very much alive. But if your reluctance results in refusal, I will have to keep working around this issue. That would be a pity since, after all, it is just three more lines of code in Core/Defs.h.

Best regards,

Tom

Subject: Re: Value with type float
Posted by mirek on Mon, 30 Oct 2023 09:34:25 GMT
View Forum Message <> Reply to Message

OK, float support (very hesitantly and conditionally) added.

https://github.com/ultimatepp/ultimatepp/blob/cb32981c04d35f061c0d700dac26e0fe546d36a7/autotest/Float/main.cpp

Some other tests I should add?

Subject: Re: Value with type float
Posted by Tom1 on Mon, 30 Oct 2023 11:27:13 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 30 October 2023 11:34OK, float support (very hesitantly and conditionally) added.

https://github.com/ultimatepp/ultimatepp/blob/cb32981c04d35f061c0d700dac26e0fe546d36a7/autotest/Float/main.cpp

Some other tests I should add?
Hi Mirek,

Thank you very, very much! Actually, the float Value support is even wider than I would have imagined... but very much appreciated. :)

No other tests come to mind.

BTW: What are the conditions and consequences of your hesitation you're referring to above?

Thanks and best regards,

Tom

EDIT: PS. In Core/Value.cpp, is it intentional or a mistake to have different return typecast for float in GetOtherInt(), GetOtherInt64(), GetOtherBool() and GetOtherDouble()?

Subject: Re: Value with type float
Posted by mirek on Mon, 30 Oct 2023 12:37:25 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 30 October 2023 12:27mirek wrote on Mon, 30 October 2023 11:34OK, float support (very hesitantly and conditionally) added.

https://github.com/ultimatepp/ultimatepp/blob/cb32981c04d35f

061c0d700dac26e0fe546d36a7/autotest/Float/main.cpp

Some other tests I should add?
Hi Mirek,

Thank you very, very much! Actually, the float Value support is even wider than I would have imagined... but very much appreciated. :)

No other tests come to mind.

BTW: What are the conditions and consequences of your hesitation you're referring to above?

If it breaks something, I rollback. But frankly it is unlikely.

Quote:
EDIT: PS. In Core/Value.cpp, is it intentional or a mistake to have different return typecast for float in GetOtherInt(), GetOtherInt64(), GetOtherBool() and GetOtherDouble()?

Thanks, fixed (it was harmless, but anyway).
