Subject: stl containers support in PDB debugger
Posted by mirek on Sun, 15 Mar 2020 17:52:57 GMT
View Forum Message <> Reply to Message

I have added some more std:: containers to "pretty" debugger display, as shown in this screenshot of all currently supported types:

File Attachments

1) Clipboard01.jpg, downloaded 1063 times

```
{
    Date date = GetSysDate();
    Time time = GetSysTime();
    Color color = LtRed();
    RGBA rgba = LtRed();
    Font fnt = Arial();
    String string = "Hello world!";
    WString wstring = "Hello world!";
    Vector<String> vector { "I", "II", "III", "IV", "V" };
    BiVector<String> bivector { "I", "II", "III", "IV", "V" };
    Array<String> array { "I", "II", "III", "IV", "V" };
    BiArray<String> biarray { "I", "II", "III", "IV", "V" };
    Index<String> index { "I", "II", "III", "IV", "V" };
    VectorMap<int, String> vectormap { { 1 , "I"}, { 2, "II" }, { 3, "III"}, { 4, "IV"}, { 5, "V
    ArrayMap<int, String> arraymap { { 1 , "I"}, { 2, "II" }, { 3, "III"}, { 4, "IV"}, { 5, "V"
    ValueArray valuearray { "I", "II", "III", "IV", "V" };
    ValueMap valuemap { { 1 , "I"}, { 2, "II" }, { 3, "III"}, { 4, "IV"}, { 5, "V" } };
    Value value = valuemap;

    std::string std_string = "Hello world!";
    std::wstring std_wstring = wstring.ToStd();
    std::vector<std::string> std_vector { "I", "II", "III", "IV", "V" };
    std::list<std::string> std_list { "I", "II", "III", "IV", "V" };
    std::set<std::string> std_set { "I", "II", "III", "IV", "V" };
    std::multiset<std::string> std_multiset { "I", "II", "III", "IV", "V" };
    std::map<int, std::string> std_map { { 1 , "I"}, { 2, "II" }, { 3, "III"}, { 4, "IV"}, { 5,
    std::multimap<int, std::string> std_multimap { { 1 , "I"}, { 2, "II" }, { 3, "III"}, { 4, "I
}
```

Autos | Locals | this | Watches | CPU | Memory | 0x3e14 | GuiMainFn_()

date	2020/03/15, raw: { day=15, month=3, year=2020 }
time	2020/03/15 18:48:07, raw: { hour=18, minute=48, second=7, day=15, month=3, year=2020 }
color	#FF0000, raw: { color=255 }
rgba	#FF0000, a: 255, raw: { b=0, g=0, r=255, a=255 }
fnt	<Arial:24>, raw: { data=144036023238658, v={ } }
string	[12] "Hello world!", raw: { chr=1ebfc48 "Hello worl", ptr=6c6c6548??, wptr=6c6c6548, qptr=6c6c654
wstring	[12] "Hello world!", raw: { len=12, s=7456350, ptr=7456350, length=12, alloc=23 }
vector	[5] "I", "II", "III", "IV", "V", raw: { vector=73cb250, items=5, alloc=5 }
bivector	[5] "I", "II", "III", "IV", "V", raw: { vector=73cb390, start=0, items=5, alloc=5 }
array	[5] "I", "II", "III", "IV", "V", raw: { vector=743e650, 743e690, 743e6d0, 743e710, 743e750 }
biarray	[5] "I", "II", "III", "IV", "V", raw: { bv=743e790, 743e810, 743e850, 743e890, 743e8d0 }
index	[5] "I", "II", "III", "IV", "V", raw: { key="I", "II", "III", "IV", "V", map=743e7d0, hash=74563b0, mask=7, un
vectormap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { key=1, 2, 3, 4, 5, value="I", "II", "III", "IV", "V" }
arraymap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { key=1, 2, 3, 4, 5, value="I", "II", "III", "IV", "V" }
valuearray	[5] "I", "II", "III", "IV", "V", raw: { vector=73c70f0, items=5, alloc=8 }, raw: { data=743ebd0 }
valuemap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { data=743ec10 }
value	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { data="", magic=1ebf4b8 }
std_string	[12] "Hello world!", raw: { __r_={ __value_={ __l={ __cap_=17, __size_=12, __data_=743ec50 "Hello wor
std_wstring	[12] "Hello world!", raw: { __r_={ __value_={ __l={ __cap_=17, __size_=12, __data_=743ed10 }, __s={ __s
std_vector	[5] "I", "II", "III", "IV", "V", raw: { __begin_=74562f0, __end_=745632c, __end_cap_={ __value_=745632c
std_list	[5] "I", "II", "III", "IV", "V", raw: { __end_={ __prev_=743ee50, __next_=743ed50 }, __size_alloc_={ __value
std_set	[5] "I", "II", "III", "IV", "V", raw: { __tree_={ __begin_node_=743ee90, __pair1_={ __value_={ __left_=743e
std_multiset	[5] "I", "II", "III", "IV", "V", raw: { __tree_={ __begin_node_=743efd0, __pair1_={ __value_={ __left_=749b
std_map	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { __tree_={ __begin_node_=749b150, __pair1_={ __value_={
std_multimap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { __tree_={ __begin_node_=749b290, __pair1_={ __value_={

Subject: Re: stl containers support in PDB debugger
Posted by mirek on Mon, 23 Mar 2020 09:12:46 GMT
View Forum Message <> Reply to Message

Added all remaining stl containers and also support for Image - that one is especially nice I think:

File Attachments

1) Clipboard01.png, downloaded 870 times

```
using namespace Upp;

GUI_APP_MAIN
{
    Date date = GetSysDate();
    Time time = GetSysTime();
    Color color = LtRed();
    RGBA rgba = LtRed();
    Font fnt = Arial();
    Image img = CtrlImg::exclamation();
    ImageBuffer ib(img);
    img = CtrlImg::exclamation();
    Value img_value = img;
    String string = "Hello world!";
    WString wstring = "Hello world!";
    Vector<String> vector { "I", "II", "III", "IV", "V" };
    BiVector<String> bivector { "I", "II", "III", "IV", "V" };
    Array<String> array { "I", "II", "III", "IV", "V" };
    BiArray<String> biarray { "I", "II", "III", "IV", "V" };
    Index<String> index { "I", "II", "III", "IV", "V" };
    VectorMap<int, String> vectormap { { 1 , "I"}, { 2, "II" }, { 3, "III"}, { 4, "IV"}, { 5, "V
```

Autos | Locals | this | Watches | CPU | Memory | 0x28d0 | GuiMainFn_()

date	2020/03/23, raw: { day=23, month=3, year=2020 }
time	2020/03/23 09:58:22, raw: { hour=9, minute=58, second=22, day=23, month=3, year=2020 }
color	#FF0000, raw: { color=255 }
rgba	#FF0000, a: 255, raw: { b=0, g=0, r=255, a=255 }
fnt	rial:24>, raw: { data=144036023238658, v={ } }
img	[64x64], raw: { data=6dee580 }
ib	[64x64], raw: { kind=0, size={ cx=64, cy=64 }, pixels={ ptr=6e51e70 }, hotspot={ x=32, y=32 }, spo
img_value	[64x64], raw: { data=6dee580 }, raw: { data="", magic=171f8f8 }
string	[12] "Hello world!", raw: { chr=171f8b8 "Hello worl", ptr=6f77206f6c6c6548??, wptr=6f77206f6c6c654
wstring	[12] "Hello world!", raw: { len=12, s=6dc6360, ptr=6dc6360, length=12, alloc=23 }
vector	[5] "I", "II", "III", "IV", "V", raw: { vector=6d8f140, items=5, alloc=5 }
bivector	[5] "I", "II", "III", "IV", "V", raw: { vector=6d8f060, start=0, items=5, alloc=5 }
array	[5] "I", "II", "III", "IV", "V", raw: { vector=6dc62a0, 6dc65a0, 6dc6480, 6dc6600, 6dc64e0 }
biarray	[5] "I", "II", "III", "IV", "V", raw: { bv=6dc6300, 6dc6720, 6dc67e0, 6dc6780, 6dc66c0 }
index	[5] "I", "II", "III", "IV", "V", raw: { key="I", "II", "III", "IV", "V", map=6dc6840, hash=6dc6660, mask=7, unlin
vectormap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { key=1, 2, 3, 4, 5, value="I", "II", "III", "IV", "V" }
arraymap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { key=1, 2, 3, 4, 5, value="I", "II", "III", "IV", "V" }
valuearray	[5] "I", "II", "III", "IV", "V", raw: { vector=6d885a0, items=5, alloc=8 }, raw: { data=6dc6ba0 }
valuemap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { data=6d72e20 }
value	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { data="", magic=171eec8 }
std_string	[12] "Hello world!", raw: { __r_={ __value_={ __l={ __cap_=8583983401233893400, __size_=14341874955
std_wstring	[12] "Hello world!", raw: { __r_={ __value_={ __l={ __cap_=17, __size_=12, __data_=6dc6cc0 }, __s={ __size
std_vector	[5] "I", "II", "III", "IV", "V", raw: { __begin_=6deab20, __end_=6deab98, __end_cap_={ __value_=6deab98 }
std_deqeue	[5] "I", "II", "III", "IV", "V", raw: { __map_={ __first_=6e60420->6e3b870, __begin_=6e60420->6e3b870, _
std_list	[5] "I", "II", "III", "IV", "V", raw: { __end_={ __prev_=6dc6ea0, __next_=6dc6d20 }, __size_alloc_={ __value_=
std_forward_list	[5] "I", "II", "III", "IV", "V", raw: { __before_begin_={ __value_={ __next_=6dc6f00 } } }
std_set	[5] "I", "II", "III", "IV", "V", raw: { __tree_={ __begin_node_=6e61120, __pair1_={ __value_={ __left_=6e611
std_multiset	[5] "I", "II", "III", "IV", "V", raw: { __tree_={ __begin_node_=6e61300, __pair1_={ __value_={ __left_=6e613
std_map	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { __tree_={ __begin_node_=6d72ea0, __pair1_={ __value_={ _
std_multimap	[5] 1: "I", 2: "II", 3: "III", 4: "IV", 5: "V", raw: { __tree_={ __begin_node_=6de91a0, __pair1_={ __value_={ _
std_unordered_set	[5] "V", "IV", "III", "II", "I", raw: { __table_={ __bucket_list_={ __ptr_={ __value_=6e61600->6e61540, __valu
std_unordered_multiset	[5] "IV", "V", "III", "II", "I", raw: { __table_={ __bucket_list_={ __ptr_={ __value_=6e61840->6e61780, __valu
std_unordered_map	[5] 5: "V", 4: "IV", 3: "III", 2: "II", 1: "I", raw: { __table_={ __bucket_list_={ __ptr_={ __value_=6e61a80->171
std_unordered_multimap	[5] 5: "V", 4: "IV", 3: "III", 2: "II", 1: "I", raw: { __table_={ __bucket_list_={ __ptr_={ __value_=6e61cc0->171

Subject: Re: stl containers support in PDB debugger
Posted by Didier on Mon, 06 Apr 2020 07:52:21 GMT
View Forum Message <> Reply to Message

Very useful, thanks !

Subject: Re: stl containers support in PDB debugger
Posted by Novo on Mon, 04 May 2020 20:48:09 GMT
View Forum Message <> Reply to Message

Neither U++ containers nor U++ strings are displayed on Linux.
Maybe I'm missing something, but full rebuild didn't fix the problem.

Subject: Re: stl containers support in PDB debugger
Posted by mirek on Mon, 04 May 2020 20:52:55 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 04 May 2020 22:48Neither U++ containers nor U++ strings are displayed on Linux.
Maybe I'm missing something, but full rebuild didn't fix the problem.

Yeah, the reason is that it only works in Windows with PDB format

I plan to start using LLDB libraries for the next release, but that is quite a lot of work...

Mirek

Subject: Re: stl containers support in PDB debugger
Posted by Novo on Mon, 04 May 2020 21:37:12 GMT
View Forum Message <> Reply to Message

I meant regular U++ containers like Vector, and regular U++ String, not STL classes ...
Basically, debugger on Linux is broken ...

Subject: Re: stl containers support in PDB debugger
Posted by mirek on Mon, 04 May 2020 22:04:17 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 04 May 2020 23:37I meant regular U++ containers like Vector, and regular U++ String, not STL classes ...
Basically, debugger on Linux is broken ...

Thats the same issue. I cannot do it with GDB... Debugger in Posix is simply the same, unchanged, as before.

Mirek