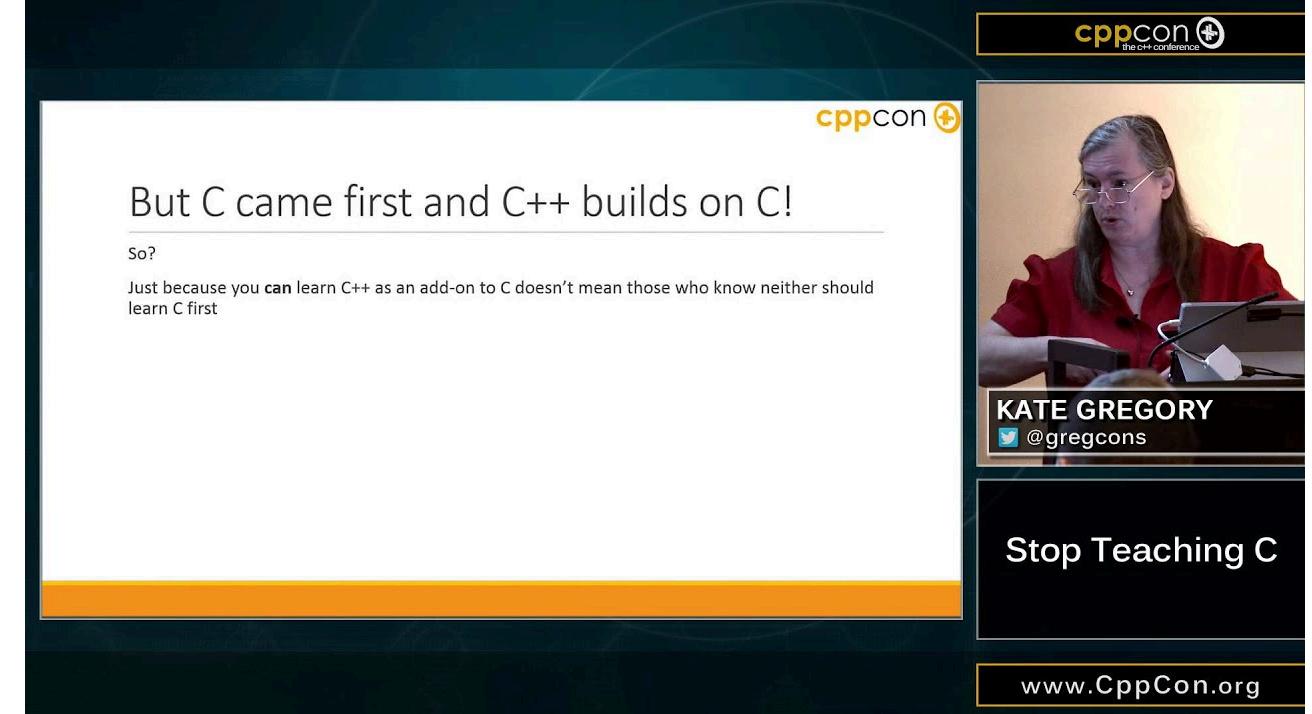


Debugger Visualizers to Make Your Code Accessible

Braden Ganetsky



"Stop Teaching C"

<https://www.youtube.com/watch?v=YnWhqhNdYyk>



Scenario: Your program hits an error and exits



Scenario: Your program hits an error and exits

- No knowledge of where it happened



Scenario: Your program hits an error and exits

- No knowledge of where it happened
- Logs are too sparse to determine what happened



Scenario: Your program hits an error and exits

- No knowledge of where it happened
- Logs are too sparse to determine what happened
- What do you do?



"printf debugging"



"printf debugging"

- **Modify the source code with a print statement**



"printf debugging"

- **Modify the source code with a print statement**
- **Recompile the program**



"printf debugging"

- **Modify the source code with a print statement**
- **Recompile the program**
- **Rerun the program, looking for the message**



"printf debugging"

- **Modify the source code with a print statement**
- **Recompile the program**
- **Rerun the program, looking for the message**
- **Realize the print statement is wrong or insufficient**



"printf debugging"

- **Modify the source code with a print statement**
- **Recompile the program**
- **Rerun the program, looking for the message**
- **Realize the print statement is wrong or insufficient**
- **Loop the above steps ad infinitum**



"printf debugging"

- **Modify the source code with a print statement**
- **Recompile the program**
- **Rerun the program, looking for the message**
- **Realize the print statement is wrong or insufficient**
- **Loop the above steps ad infinitum**
- **All the while, recompiling on every loop**



"printf debugging"

- **Modify the source code with a print statement**
- **Recompile the program**
- **Rerun the program, looking for the message**
- **Realize the print statement is wrong or insufficient**
- **Loop the above steps ad infinitum**
- **All the while, recompiling on every loop**
- **Big waste of time**



Possible alternatives

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 11 11 11 11
10 00 11 01 10 01 0
00 11 11 11 11 11 1
01 10 01 10 01 10 01 0
01 01 01 01 01 01 01 0
11 00 00 11 00 00 11 00 0
09 11 11 09 11 11 09 11 1
01 1 01 1 01 1 01 1 01 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1



Possible alternatives

- **Don't write bugs?**



Possible alternatives

- **Don't write bugs?**
- **Use sanitizers to reduce bugs?**



Possible alternatives

- **Don't write bugs?**
- **Use sanitizers to reduce bugs?**
- **Others...?**



Possible alternatives

- **Don't write bugs?**
- **Use sanitizers to reduce bugs?**
- **Others...?**
- **Use a debugger to explore and walk through the code!**



Using a debugger: The pros



Using a debugger: The pros

- No recompiling



Using a debugger: The pros

- No recompiling
- Iterating is much faster



Using a debugger: The pros

- No recompiling
- Iterating is much faster
- As granular as you want, e.g. stepping into function calls



Using a debugger: The pros

- No recompiling
- Iterating is much faster
- As granular as you want, e.g. stepping into function calls
- Special features, e.g. time travel debugging



Using a debugger: The cons



Using a debugger: The cons

- More painful to set up



Using a debugger: The cons

- More painful to set up
- Learning curve



Using a debugger: The cons

- More painful to set up
- Learning curve
- Less accessible to get started



Using a debugger: The cons

- More painful to set up
- Learning curve
- Less accessible to get started
- Incomprehensible data displayed



Incomprehensible data

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 11 11 11 11 11



Incomprehensible data

```
#include <boost/unordered_set.hpp>
int main() {
    boost::unordered_set<int> set{ 11, 22, 33 };
}
```



Incomprehensible data

```
#include <boost/unordered_set.hpp>
int main() {
    boost::unordered_set<int> set{ 11, 22, 33 };
}
```

▲ 🔑 set	{table_= {size_=3 mlf_=1.00000000 ...
▲ 🔑 table_	{size_=3 mlf_=1.00000000 max_loa...
▲ 🔑 boost::un...	{current_=0 '\0' funcs_=0x000000b0...
▲ 🔑 current_	0 '\0'
▶ 🔑 funcs_	0x000000b07b7cf401 {{t_=equal_to ...
▲ 🔑 size_	3
▲ 🔑 mlf_	1.00000000
▲ 🔑 max_load_	13
▶ 🔑 buckets_	allocator



Incomprehensible data

```
#include <boost/unordered_set.hpp>
int main() {
    boost::unordered_set<int> set{ 11, 22, 33 };
}
```

◀	eraser	set	{ size=3 }
▶	cube	[hash_function]	{...}
▶	cube	[key_eq]	equal_to
▶	eraser	[allocator]	allocator
	eraser	[0]	33
	eraser	[1]	22
	eraser	[2]	11
▶	eraser	[Raw View]	{table_= {size_=3 mlf_=1.00000000 ...}



Incomprehensible data

```
#include <boost/unordered_set.hpp>
int main() {
    boost::unordered_set<int> set{ 11, 22, 33 };
}

(gdb) print set
$1 = {
  table_ = {
    <boost::unordered::detail::functions<boost::hash<int>, std::equal_to<int> >> = {
      static nothrow_move_assignable = true,
      static nothrow_move_constructible = true,
      static nothrow_swappable = true,
      current_ = 0 '\000',
      funcs_ = {{
        t_ = {
          <boost::unordered::detail::compressed_base<boost::hash<int>, 1>> = {
            <boost::empty::empty_value<boost::hash<int>, 0u, true>> = {
              <boost::hash<int>> = {<No data fields>}, <No data fields>}, <No data fields>},
            <boost::unordered::detail::compressed_base<std::equal_to<int>, 2>> = {
              <boost::empty::empty_value<std::equal_to<int>, 0u, true>> = {
                <std::equal_to<int>> = {
                  <std::binary_function<int, int, bool>> = {<No data fields>}, <No data fields>}, <No data fields>},
                },
              },
            },
          },
        t_ = {
          <boost::unordered::detail::compressed_base<boost::hash<int>, 1>> = {
            <boost::empty::empty_value<boost::hash<int>, 0u, true>> = {
              <boost::hash<int>> = {<No data fields>}, <No data fields>}, <No data fields>},
            <boost::unordered::detail::compressed_base<std::equal_to<int>, 2>> = {
              <boost::empty::empty_value<std::equal_to<int>, 0u, true>> = {
                
```

```
<std::equal_to<int>> = {
    <std::binary_function<int, int, bool>> = {<No data fields>, <No data fields>, <No data fields>, <No data fields>}
}
},
members of boost::unordered::detail::table<boost::unordered::detail::set<std::allocator<int>, int, boost::hash<int>, std::equal_to<
size_ = 3,
mlf_ = 1,
max_load_ = 13,
buckets_ = {
    <boost::empty_::empty_value<std::allocator<boost::unordered::detail::node<int, void*>>, 0u, true>> = {
        <std::allocator<boost::unordered::detail::node<int, void*>>> = {
            <__gnu_cxx::new_allocator<boost::unordered::detail::node<int, void*>>> = {<No data fields>, <No data fields>, <No data fields>}
members of boost::unordered::detail::grouped_bucket_array<boost::unordered::detail::bucket<boost::unordered::detail::node<int, void*>>
size_index_ = 0,
size_ = 13,
buckets = 0x55555556feb0,
groups = 0x55555556ff30
}
}
}
}
```

Incomprehensible data

```
#include <boost/unordered_set.hpp>
int main() {
    boost::unordered_set<int> set{ 11, 22, 33 };
}
```

```
(gdb) print set
$1 = boost::unordered_set with 3 elements = {
    [0] = 33,
    [1] = 22,
    [2] = 11
}
```



In this talk



In this talk

- Addressing the "cons" of debuggers



In this talk

- Addressing the "cons" of debuggers
- Advocating for writing code with tooling and users in mind



In this talk

- Addressing the "cons" of debuggers
- Advocating for writing code with tooling and users in mind
- The "how" of Visual Studio Natvis



In this talk

- Addressing the "cons" of debuggers
- Advocating for writing code with tooling and users in mind
- The "how" of Visual Studio Natvis
- The "how" of GDB pretty-printers



In this talk

- Addressing the "cons" of debuggers
- Advocating for writing code with tooling and users in mind
- The "how" of Visual Studio Natvis
- The "how" of GDB pretty-printers
- Barely any C++



In this talk

- Addressing the "cons" of debuggers
- Advocating for writing code with tooling and users in mind
- The "how" of Visual Studio Natvis
- The "how" of GDB pretty-printers
- Barely any C++
- Questions at any time!



About me



About me

- Canadian



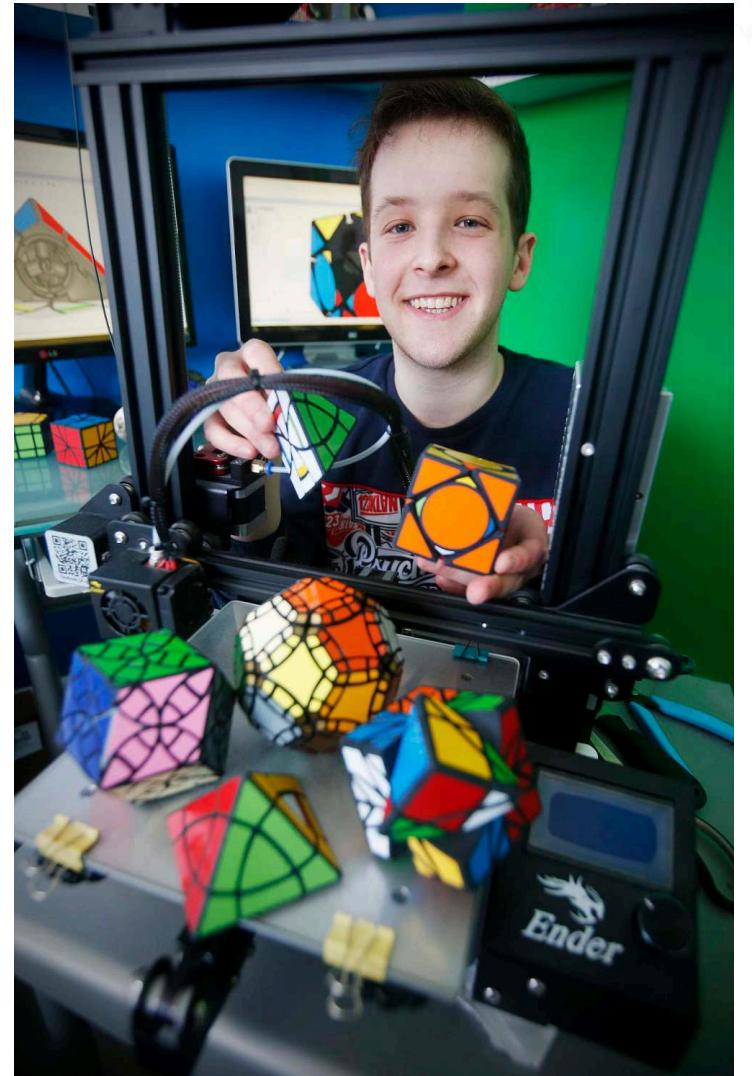
About me

- Canadian
- Mechanical engineering degree



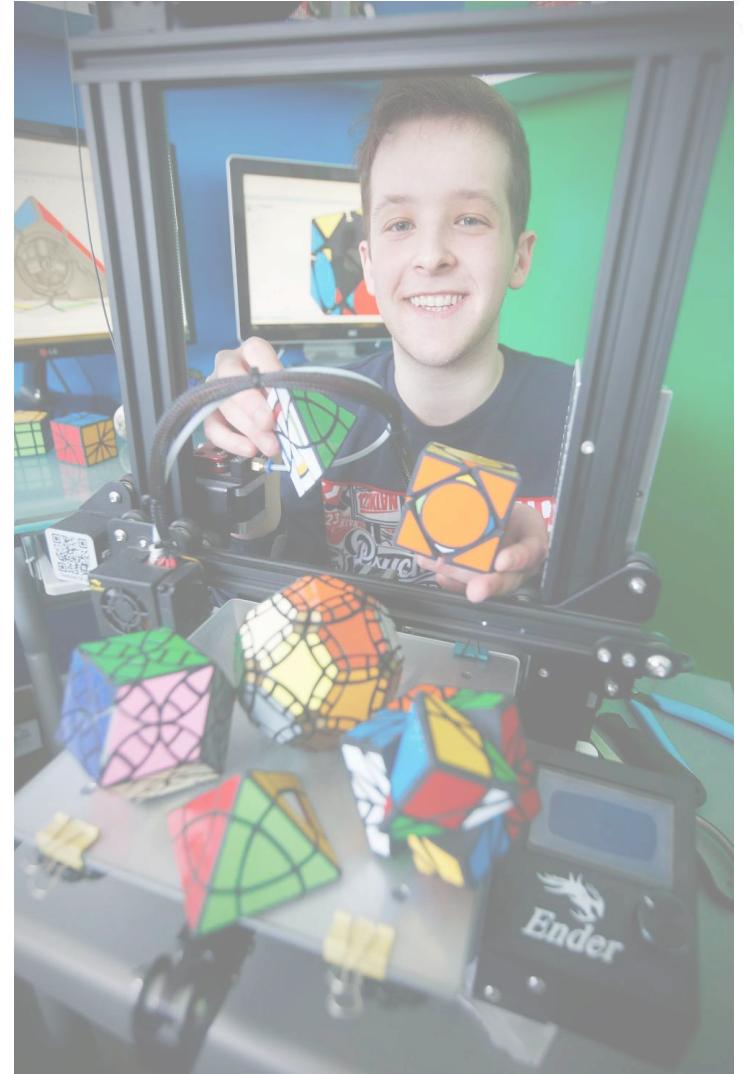
About me

- Canadian
- Mechanical engineering degree
- I like twisty puzzles



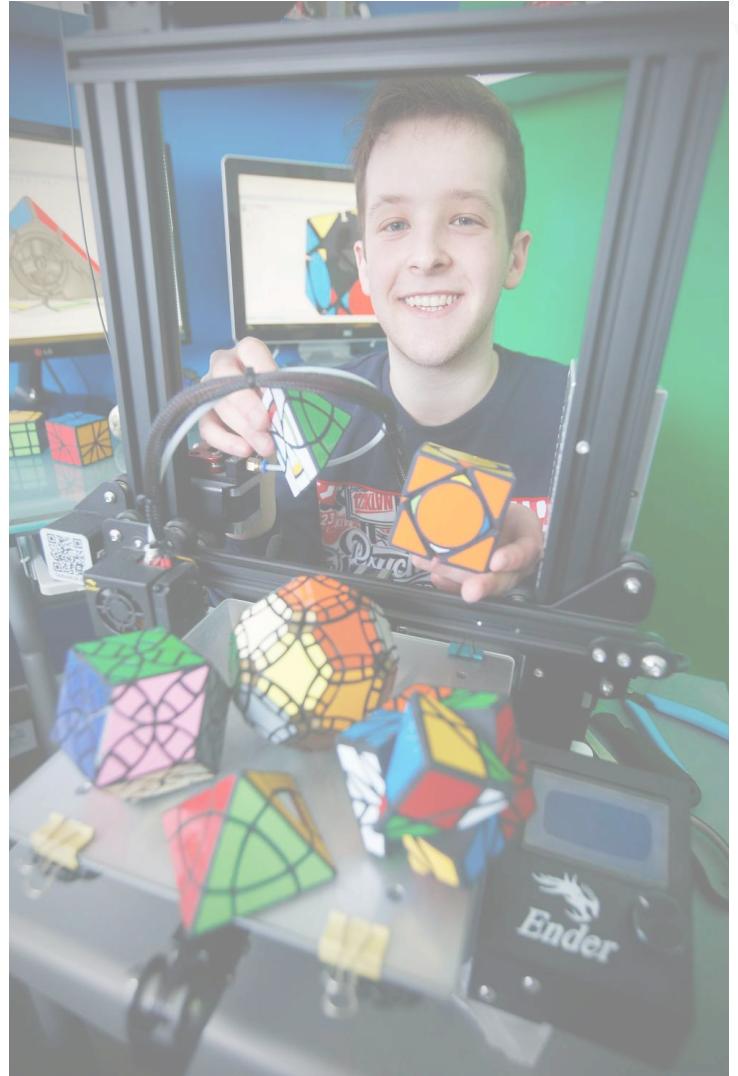
About me

- Canadian
- Mechanical engineering degree
- I like twisty puzzles
- C++ standards committee (WG21)



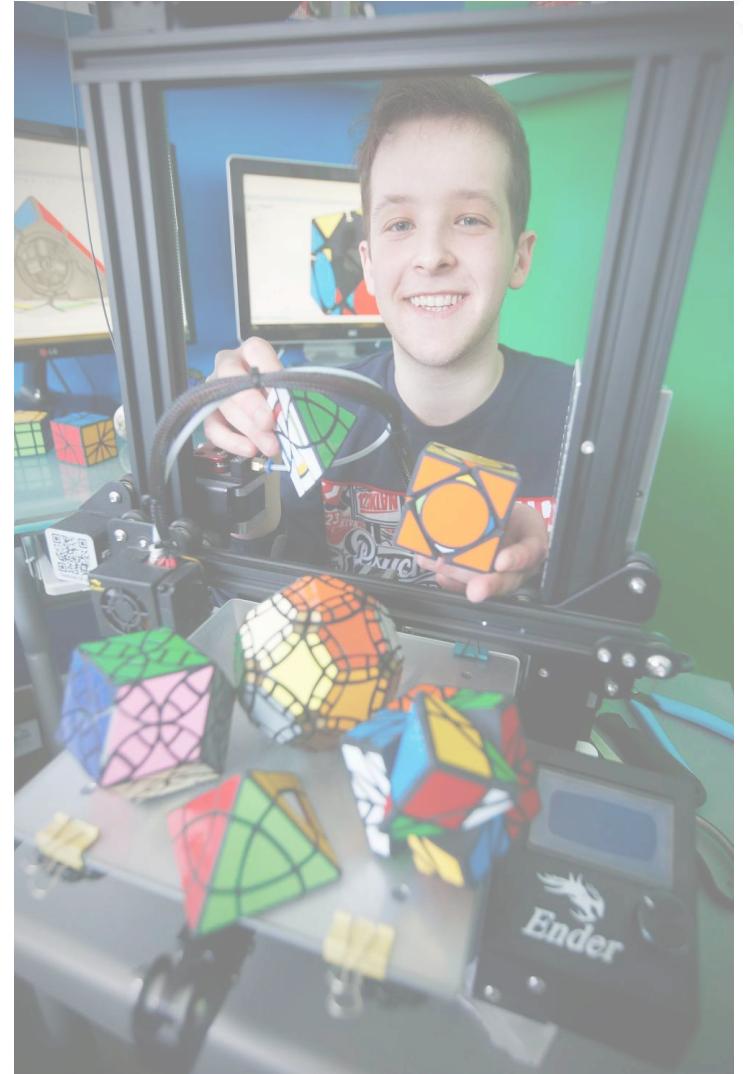
About me

- Canadian
- Mechanical engineering degree
- I like twisty puzzles
- C++ standards committee (WG21)
- Winnipeg C++ Developers meet up group



About me

- Canadian
- Mechanical engineering degree
- I like twisty puzzles
- C++ standards committee (WG21)
- Winnipeg C++ Developers meet up group
- CppNorth volunteer coordinator



Natvis setup

Natvis files



Natvis files

- XML



Natvis files

- XML
- **.natvis file extension**



Natvis files

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

</AutoVisualizer>
```



Natvis files

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

    <Type Name="MyClass">
        ...
    </Type>

</AutoVisualizer>
```



Natvis files

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

    <Type Name="MyClass">
        ...
    </Type>

    <Type Name="MyClassTemplate<*>">
        ...
    </Type>

</AutoVisualizer>
```



Set up Natvis



Set up Natvis

- In Visual Studio

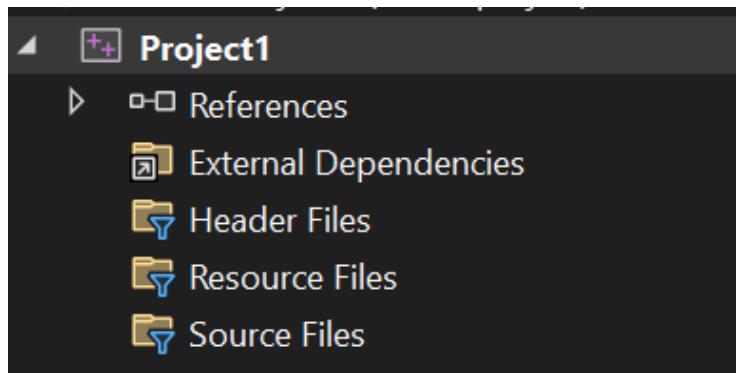


Set up Natvis

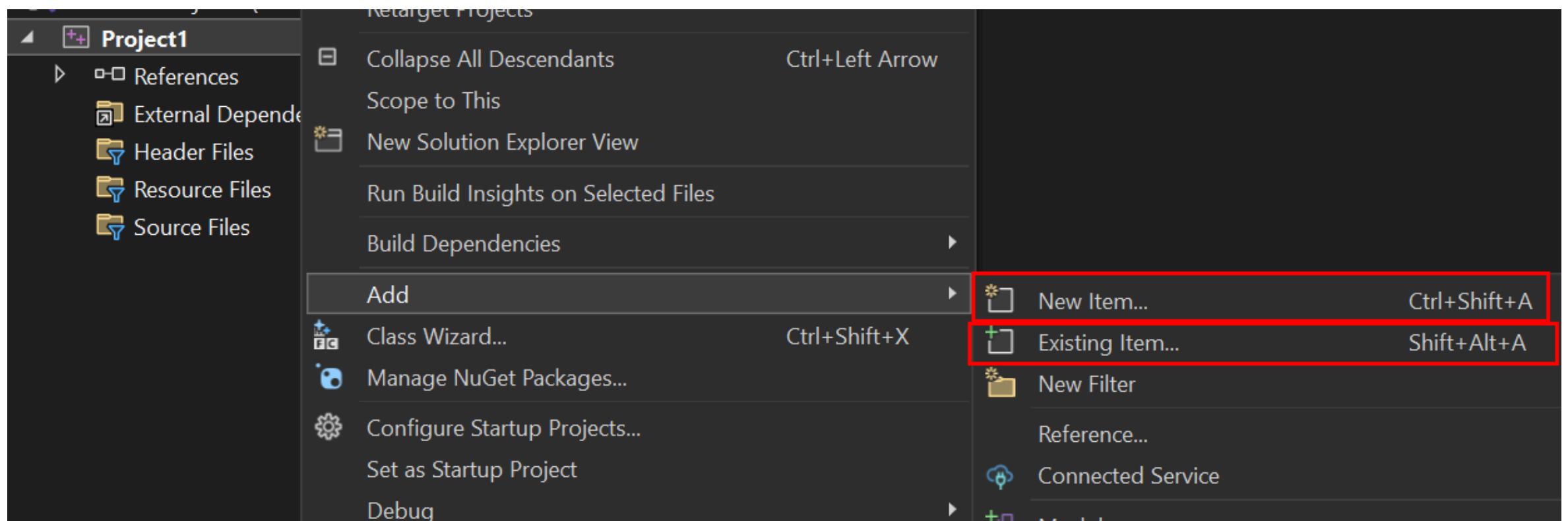
- In Visual Studio
- In CMake



In Visual Studio



In Visual Studio



In Visual Studio

Add New Item - Project1

Sort by: Default

Installed

Visual C++

- Code
- Formatting
- ATL
- Data
- Resource
- Web
- Utility
- Property Sheets
- Test
- HLSL
- Graphics

Online

	Registration Script (.rgs)	Visual C++
	Code Analysis Rule Set	Visual C++
	Text File (.txt)	Visual C++
	Class Diagram	Visual C++
	Debugger visualization file (.natvis)	Visual C++
	Markdown file	Visual C++



In CMake



In CMake

```
add_executable(my_target
    # ... all other args
    path/to/my.natvis
)
```



In CMake

```
add_executable(my_target
    # ... all other args
    path/to/my.natvis
)
```

```
target_sources(my_target
    PUBLIC path/to/my.natvis
)
```



Results



Results

```
struct Line {  
    int left;  
    int right;  
};
```



Results

```
struct Line {  
    int left;  
    int right;  
};
```

```
Line line{ 100, 600 };
```



Results

```
struct Line {  
    int left;  
    int right;  
};
```

```
Line line{ 100, 600 };
```

Locals		
Search (Ctrl+E)		Search Depth
Name	Value	Type
line	{left=100 right=600 }	Line
left	100	int
right	600	int



Results

```
struct Line {  
    int left;  
    int right;  
};
```

```
<Type Name="Line">  
    <DisplayString>{{left}, {right}}</DisplayString>  
    <Expand>  
        <Item Name="[length]">right - left</Item>  
    </Expand>  
</Type>
```

```
Line line{ 100, 600 };
```

Locals		
Search (Ctrl+E)		Search Depth
Name	Value	Type
line	{left=100 right=600 }	Line
left	100	int
right	600	int



Results

```
struct Line {
    int left;
    int right;
};
```

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand>
        <Item Name="[length]">right - left</Item>
    </Expand>
</Type>
```

```
Line line{ 100, 600 };
```

Locals		
Search (Ctrl+E) Search Depth		
Name	Value	Type
line	{left=100 right=600 }	Line
left	100	int
right	600	int

Locals		
Search (Ctrl+E) Search Depth		
Name	Value	Type
line	{100, 600}	Line
[length]	500	int
[Raw View]	{left=100 right=600 }	Line
left	100	int
right	600	int



Writing Natvis

Natvis <Type> tag

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

    <Type Name="MyClass">
        ...
    </Type>

    <Type Name="MyClassTemplate<*>">
        ...
    </Type>

</AutoVisualizer>
```



Natvis <Type> tag

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

    <Type Name="MyClass">
        ...
    </Type>

    <Type Name="MyClassTemplate<*>">
        ...
    </Type>

</AutoVisualizer>
```



Natvis <Type> tag

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

    <Type Name="MyClass">
        ...
    </Type>

    <Type Name="MyClassTemplate<*>">
        ...
    </Type>

</AutoVisualizer>
```



Inside the Natvis <Type> tag



Inside the Natvis <Type> tag

- <AlternativeType> tag, and attributes on <Type>



Inside the Natvis <Type> tag

- <AlternativeType> tag, and attributes on <Type>
- <DisplayString> tag



Inside the Natvis <Type> tag

- <AlternativeType> tag, and attributes on <Type>
- <DisplayString> tag
- **Expansions**



Inside the Natvis <Type> tag

- <AlternativeType> tag, and attributes on <Type>
- <DisplayString> tag
- Expansions
- <Intrinsic> tag



Importantly



Importantly

- Any data members, no matter the access specifiers



Importantly

- Any data members, no matter the access specifiers
- **No user-defined C++ functions**



Importantly

- Any data members, no matter the access specifiers
- No user-defined C++ functions
- Some Natvis features are not documented



Importantly

- Any data members, no matter the access specifiers
- No user-defined C++ functions
- Some Natvis features are not documented
- This presentation is not exhaustive



<AlternativeType> tag



<AlternativeType> tag

```
<!-- STL.natvis -->
<Type Name="std::unordered_map*>" Priority="Medium">
    <AlternativeType Name="std::unordered_multimap*>" />
    <AlternativeType Name="std::unordered_set*>" />
    <AlternativeType Name="std::unordered_multiset*>" />
    ...
</Type>
```



<AlternativeType> tag

```
<!-- STL.natvis -->
<Type Name="std::unordered_map*>" Priority="Medium">
    <AlternativeType Name="std::unordered_multimap*>" />
    <AlternativeType Name="std::unordered_set*>" />
    <AlternativeType Name="std::unordered_multiset*>" />
    ...
</Type>
```



<AlternativeType> tag

```
<!-- STL.natvis -->
<Type Name="std::unordered_map*>" Priority="Medium">
    <AlternativeType Name="std::unordered_multimap*>" />
    <AlternativeType Name="std::unordered_set*>" />
    <AlternativeType Name="std::unordered_multiset*>" />
    ...
</Type>
```



<AlternativeType> tag

```
<!-- STL.natvis -->
<Type Name="std::unordered_map*>" Priority="Medium">
    <AlternativeType Name="std::unordered_multimap*>" />
    <AlternativeType Name="std::unordered_set*>" />
    <AlternativeType Name="std::unordered_multiset*>" />
    ...
</Type>
```



Inheritable attribute



Inheritable attribute

```
class B {  
    // ...  
};
```



Inheritable attribute

```
class B {  
    // ...  
};
```

```
class D : public B {  
    // ...  
};
```



Inheritable attribute

```
class B {  
    // ...  
};  
  
class D : public B {  
    // ...  
};
```

```
<Type Name="B" Inheritable="false">  
    ...  
</Type>
```



Inheritable attribute

```
class B {  
    // ...  
};  
  
class D : public B {  
    // ...  
};
```

```
<Type Name="B" Inheritable="false">  
    ...  
</Type>
```

Note: defaults to "true"



IncludeView and ExcludeView



IncludeView and ExcludeView

```
<Type Name="MyClass" IncludeView="foo">  
    ...  
</Type>
```



IncludeView and ExcludeView

```
<Type Name="MyClass" IncludeView="foo">  
    ...  
</Type>
```

```
<Type Name="MyClass" ExcludeView="bar">  
    ...  
</Type>
```



IncludeView and ExcludeView

```
<Type Name="MyClass" IncludeView="foo">  
    ...  
</Type>  
  
<Type Name="MyClass" ExcludeView="bar">  
    ...  
</Type>
```

Watch 1

Search (Ctrl+E)

Name	Value
my_class	{...}
my_class,view(foo)	{...}
my_class,view(bar)	{...}
Add item to watch	



Priority attribute



Priority attribute

```
<Type Name="MyClass" Priority="Medium">  
    ...  
</Type>
```



Priority attribute

```
<Type Name="MyClass" Priority="Medium">  
    ...  
</Type>
```

```
<Type Name="MyClass" Priority="MediumHigh">  
    ...  
</Type>
```



Priority with views



Priority with views

```
<Type Name="std::unordered_map*>" Priority="Medium">  
    ...  
</Type>
```



Priority with views

```
<Type Name="std::unordered_map*>" Priority="Medium">
    ...
</Type>

<Type Name="std::unordered_map*>" Priority="MediumHigh"
      ExcludeView="ShowElementsByIndex"
    >
    ...
</Type>
```



<DisplayString> tag



<DisplayString> tag

A format string



<DisplayString> tag

A format string

```
<Type Name="Point">
    <DisplayString>{{x={x}, y={y}}}</DisplayString>
</Type>
```



<DisplayString> tag

A format string

```
<Type Name="Point">
    <DisplayString>{{x={x}, y={y}}}</DisplayString>
</Type>
```

Name	Value
point	{x=3, y=5}



<DisplayString> tag



<DisplayString> tag

```
<!-- STL.natvis -->
<Type Name="std::optional<*>">
    <DisplayString Condition="!_Has_value">nullopt</DisplayString>
    <DisplayString Condition="_Has_value">{_Value}</DisplayString>
    ...
</Type>
```



<DisplayString> tag

```
<!-- STL.natvis -->
<Type Name="std::optional<*>">
    <DisplayString Condition="!_Has_value">nullopt</DisplayString>
    <DisplayString Condition="_Has_value">{_Value}</DisplayString>
    ...
</Type>
```



<DisplayString> tag

```
<!-- STL.natvis -->
<Type Name="std::optional<*>">
    <DisplayString Condition="!_Has_value">nullopt</DisplayString>
    <DisplayString Condition="_Has_value">{_Value}</DisplayString>
    ...
</Type>
```



<DisplayString> tag

```
<!-- STL.natvis -->
<Type Name="std::optional<*>">
    <DisplayString Condition="!_Has_value">nullopt</DisplayString>
    <DisplayString Condition="_Has_value">{_Value}</DisplayString>
    ...
</Type>
```



<DisplayString> tag

```
<!-- STL.natvis -->
<Type Name="std::optional<*>">
    <DisplayString Condition="!_Has_value">nullopt</DisplayString>
    <DisplayString Condition="_Has_value">{_Value}</DisplayString>
    ...
</Type>
```



Basic expansions



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand>
        <Item Name="[length]">right - left</Item>

    </Expand>
</Type>
```



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{{left}, {right}}}</DisplayString>
    <Expand>
        <Item Name="[length]">right - left</Item>

    </Expand>
</Type>
```



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand>
        <Item Name="[length]">right - left</Item>
    </Expand>
</Type>
```

▶	line	{100, 600}
	[length]	500
▶	[Raw View]	{left=100 right=600 }



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand>
        <Item Name="[length]">right - left</Item>
        <Item Name="dummy">123</Item>
        <Item Name="(other dummy)">456 + left</Item>
    </Expand>
</Type>
```

▶	line	{100, 600}
	[length]	500
	dummy	123
	(other dummy)	556
▶	[Raw View]	{left=100 right=600 }



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand>
        <Item Name="[length]">right - left</Item>
        <Item Name="dummy">123</Item>
        <Item Name="(other dummy)">456 + left</Item>
    </Expand>
</Type>
```

▶	line	{100, 600}
	[length]	500
	dummy	123
	(other dummy)	556
▶	[Raw View]	{left=100 right=600 }



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <Item Name="dummy">123</Item>
        <Item Name="(other dummy)">456 + left</Item>
    </Expand>
</Type>
```

line	{100, 600}
[length]	500
dummy	123
(other dummy)	556



Basic expansions

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <Item Name="dummy">123</Item>
        <Item Name="(other dummy)">456 + left</Item>
    </Expand>
</Type>
```

line	{100, 600}
[length]	500
dummy	123
(other dummy)	556



<Item> tag

```
<Item Name="[length]">right - left</Item>
```



<Item> tag

```
<Item  
    Name="[length]"  
  
>  
    right - left  
</Item>
```



<Item> tag

```
<Item  
    Name="[length]"  
    Condition="right - left &gt;= 0"  
  
>  
    right - left  
</Item>
```



<Item> tag

```
<Item
    Name="[length]"
    Condition="right - left &gt;= 0"
    IncludeView="detailed;advanced"
>
    right - left
</Item>
```



<Item> tag

```
<Item
    Name="[length]"
    Condition="right - left &gt;= 0"
    ExcludeView="simple"
>
    right - left
</Item>
```



<Item> tag

```
<Item  
    Name="[length]"  
    Condition="right - left &gt;= 0"  
    ExcludeView="simple"  
>  
    right - left  
</Item>
```



<ArrayItems> tag



<ArrayItems> tag

```
<!-- STL.natvis -->
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>

        </Expand>
    </Type>
```



<ArrayItems> tag

```
<!-- STL.natvis -->
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>

        </Expand>
    </Type>
```



<ArrayItems> tag

```
<!-- STL.natvis -->
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>

        </Expand>
    </Type>
```



<ArrayItems> tag

```
<!-- STL.natvis -->
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>

        </Expand>
    </Type>
```



<ArrayItems> tag

```
<!-- STL.natvis -->
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>
        <ArrayItems>
            <Size>$T2</Size>
            <ValuePointer>_Elems</ValuePointer>
        </ArrayItems>
    </Expand>
</Type>
```



<ArrayItems> tag

```
<!-- STL.natvis -->
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>
        <ArrayItems>
            <Size>$T2</Size>
            <ValuePointer>_Elems</ValuePointer>
        </ArrayItems>
    </Expand>
</Type>
```



<ArrayItems> tag

```
<Type Name="std::array<*,*>">
  <DisplayString>{{ size={$T2} }}</DisplayString>
  <Expand>
    <ArrayItems>
      <Size>$T2</Size>
      <ValuePointer>_Elems</ValuePointer>
    </ArrayItems>
  </Expand>
</Type>
```



<ArrayItems> tag

```
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>
        <ArrayItems>
            <Size>$T2</Size>
            <ValuePointer>_Elems</ValuePointer>
        </ArrayItems>
    </Expand>
</Type>
```

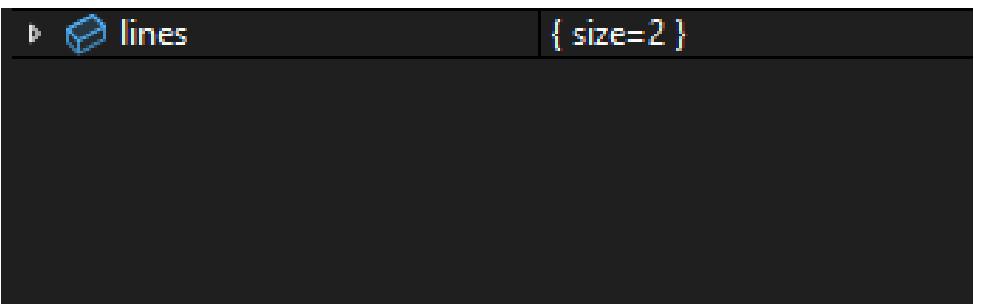
```
std::array<Line, 2> lines{
    Line{100, 600},
    Line{2, 5},
};
```



<ArrayItems> tag

```
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>
        <ArrayItems>
            <Size>$T2</Size>
            <ValuePointer>_elems</ValuePointer>
        </ArrayItems>
    </Expand>
</Type>
```

```
std::array<Line, 2> lines{
    Line{100, 600},
    Line{2, 5},
};
```



<ArrayItems> tag

```
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>
        <ArrayItems>
            <Size>$T2</Size>
            <ValuePointer>_Elems</ValuePointer>
        </ArrayItems>
    </Expand>
</Type>
```

```
std::array<Line, 2> lines{
    Line{100, 600},
    Line{2, 5},
};
```

▲	lines	{ size=2 }
▶	[0]	{100, 600}
▶	[1]	{2, 5}
▶	[Raw View]	{_Elems=0x000000b0ec15f...}



<ArrayItems> tag

```
<Type Name="std::array<*,*>">
    <DisplayString>{{ size={$T2} }}</DisplayString>
    <Expand>
        <ArrayItems>
            <Size>$T2</Size>
            <ValuePointer>_elems</ValuePointer>
        </ArrayItems>
    </Expand>
</Type>
```

```
std::array<Line, 2> lines{
    Line{100, 600},
    Line{2, 5},
};
```

lines	{ size=2 }
[0]	{100, 600}
[length]	500
[1]	{2, 5}
[length]	3
[Raw View]	{_elems=0x000000b0ec15f...}



Other "automatic" expansions

1
10
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 11 11 11 11 11



Other "automatic" expansions

<IndexListItems>



Other "automatic" expansions

<IndexListItems>

- **Expand sequences based on an index**



Other "automatic" expansions

<IndexListItems>

- Expand sequences based on an index

<LinkedListItems>



Other "automatic" expansions

<IndexListItems>

- **Expand sequences based on an index**

<LinkedListItems>

- **Expand linked list structures**



Other "automatic" expansions

<IndexListItems>

- Expand sequences based on an index

<LinkedListItems>

- Expand linked list structures

<TreeItems>



Other "automatic" expansions

<IndexListItems>

- **Expand sequences based on an index**

<LinkedListItems>

- **Expand linked list structures**

<TreeItems>

- **Expand certain shapes of tree structures**



<ExpandedItem> tag



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```

```
<Type Name="LineWrapper">  
    <DisplayString>{line}</DisplayString>  
    <Expand>  
        <ExpandedItem>line</ExpandedItem>  
    </Expand>  
</Type>
```



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```

```
<Type Name="LineWrapper">  
    <DisplayString>{line}</DisplayString>  
    <Expand>  
        <ExpandedItem>line</ExpandedItem>  
    </Expand>  
</Type>
```



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```

```
<Type Name="LineWrapper">  
    <DisplayString>{line}</DisplayString>  
    <Expand>  
        <ExpandedItem>line</ExpandedItem>  
    </Expand>  
</Type>
```



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```

```
<Type Name="LineWrapper">  
    <DisplayString>{line}</DisplayString>  
    <Expand>  
        <ExpandedItem>line</ExpandedItem>  
    </Expand>  
</Type>
```



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```

```
<Type Name="LineWrapper">  
    <DisplayString>{line}</DisplayString>  
    <Expand>  
        <ExpandedItem>line</ExpandedItem>  
    </Expand>  
</Type>
```

```
Line line{ 100, 600 };  
LineWrapper lw{ line };
```



<ExpandedItem> tag

```
struct LineWrapper {  
    Line line;  
};
```

```
<Type Name="LineWrapper">  
    <DisplayString>{line}</DisplayString>  
    <Expand>  
        <ExpandedItem>line</ExpandedItem>  
    </Expand>  
</Type>
```

```
Line line{ 100, 600 };  
LineWrapper lw{ line };
```

▲	line	{100, 600}
	[length]	500
▲	lw	{100, 600}
	[length]	500
▶	[Raw View]	{line={100, 600} }



<Synthetic> tag



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
    </Expand>
</Type>
```



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
    </Expand>
</Type>
```

```
Line line{ 100, 600 };
```



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
    </Expand>
</Type>
```

```
Line line{ 100, 600 };
```

line	{100, 600}
[left]	100
[right]	600



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
        <Synthetic Name="[calculated]">
            </Synthetic>
        </Expand>
    </Type>
```

```
Line line{ 100, 600 };
```

line	{100, 600}
[left]	100
[right]	600
[calculated]	



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
        <Synthetic Name="[calculated]">
            <DisplayString>Line properties</DisplayString>
        </Synthetic>
    </Expand>
</Type>
```

```
Line line{ 100, 600 };
```

line	{100, 600}
[left]	100
[right]	600
[calculated]	Line properties



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
        <Synthetic Name="[calculated]">
            <DisplayString>Line properties</DisplayString>
            <Expand>
                <Item Name="[length]">right - left</Item>
            </Expand>
        </Synthetic>
    </Expand>
</Type>
```

```
Line line{ 100, 600 };
```

line	{100, 600}
[left]	100
[right]	600
[calculated]	Line properties
[length]	500



<Synthetic> tag

```
<Type Name="Line">
    <DisplayString>{{left}, {right}}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[left]">left</Item>
        <Item Name="[right]">right</Item>
        <Synthetic Name="[calculated]">
            <DisplayString>Line properties</DisplayString>
            <Expand>
                <Item Name="[length]">right - left</Item>
                <Item Name="[mid]">(left + right)/2</Item>
            </Expand>
        </Synthetic>
    </Expand>
</Type>
```

```
Line line{ 100, 600 };
```

line	{100, 600}
[left]	100
[right]	600
[calculated]	Line properties
[length]	500
[mid]	350



<CustomListItems> tag



<CustomListItems> tag

- A "manual" expansion



<CustomListItems> tag

- A "manual" expansion
- **Create variables**



<CustomListItems> tag

- A "manual" expansion
- Create variables
- Execute statements



<CustomListItems> tag

- A "manual" expansion
- Create variables
- Execute statements
- Branches and loops



<CustomListItems> tag

- A "manual" expansion
- Create variables
- Execute statements
- Branches and loops
- Output items



<CustomListItems> tag

- A "manual" expansion
- Create variables
- Execute statements
- Branches and loops
- Output items
- **Basically a DSL to iterate over your structure**



<CustomListItems> tag



<CustomListItems> tag

```
<Type Name="Line">
    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<CustomListItems> tag

```
<Type Name="Line">
    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<CustomListItems> tag

```
<Type Name="Line">
    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            ...
        </CustomListItems>
    </Expand>
</Type>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
```

```
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
```

```
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)" />
    ...
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>

        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
        <If Condition="current > right">
            <Break/>
        </If>
    </Loop>
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>

        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
        <If Condition="current > right">
            <Break/>
        </If>
    </Loop>
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>

        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
        <If Condition="current > right">
            <Break/>
        </If>
    </Loop>
</CustomListItems>
```

line	{ length=500 }
[length]	500



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
        <If Condition="current > right">
            <Break/>
        </If>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
        <If Condition="current > right">
            <Break/>
        </If>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>

        <Break Condition="current > right"/>

    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop>
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>

        <Break Condition="current > right"/>

    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop Condition="current <= right">
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop Condition="current <= right">
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)" />
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Loop Condition="current <= right">
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Item Name="[steps]">steps</Item>
    <Loop Condition="current <= right">
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[steps]	5
	[1]	100
	[2]	225
	[3]	350
	[4]	475
	[5]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)" />
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Item Name="[steps]">steps</Item>
    <Loop Condition="current &lt;= right">
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[steps]	5
	[1]	100
	[2]	225
	[3]	350
	[4]	475
	[5]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Item Name="[steps]">steps</Item>
    <Loop Condition="current <= right">
        <Item>current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[steps]	5
	[1]	100
	[2]	225
	[3]	350
	[4]	475
	[5]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" initialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)"/>
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Item Name="[steps]">steps</Item>
    <Loop Condition="current <= right">
        <Item Name="[{index}]">current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[steps]	5
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<CustomListItems> tag

```
<CustomListItems>
    <Variable Name="steps" InitialValue="5"/>
    <Variable Name="step_size"
        initialValue="(right - left) / (steps - 1)" />
    <Variable Name="current" initialValue="left"/>
    <Variable Name="index" initialValue="0"/>

    <Item Name="[steps]">steps</Item>
    <Loop Condition="current &lt;= right">
        <Item Name="[{index}]">current</Item>
        <Exec>++index</Exec>
        <Exec>current = left + index * step_size</Exec>
    </Loop>
</CustomListItems>
```

	line	{ length=500 }
	[length]	500
	[steps]	5
	[0]	100
	[1]	225
	[2]	350
	[3]	475
	[4]	600



<Intrinsic> tag



<Intrinsic> tag

- User-defined functions within Natvis



<Intrinsic> tag

- User-defined functions within Natvis
- Member function if within <Type> tag



<Intrinsic> tag

- User-defined functions within Natvis
- Member function if within <Type> tag
- Free function if outside <Type> tags



<Intrinsic> tag

- User-defined functions within Natvis
- Member function if within <Type> tag
- Free function if outside <Type> tags
- Can have parameters



<Intrinsic> tag

- User-defined functions within Natvis
- Member function if within <Type> tag
- Free function if outside <Type> tags
- Can have parameters
- Can be overloaded



<Intrinsic> tag

```
<Type Name="Line">

    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            <Variable Name="steps" InitialValue="5"/>
            <Variable Name="step_size"
                initialValue="(right - left) / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<Intrinsic> tag

```
<Type Name="Line">

    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            <Variable Name="steps" initialValue="5"/>
            <Variable Name="step_size"
                initialValue="(right - left) / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<Intrinsic> tag

```
<Type Name="Line">
    <Intrinsic Name="length" />
    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            <Variable Name="steps" InitialValue="5"/>
            <Variable Name="step_size"
                initialValue="(right - left) / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<Intrinsic> tag

```
<Type Name="Line">
    <Intrinsic Name="length" Expression="right - left"/>
    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            <Variable Name="steps" initialValue="5"/>
            <Variable Name="step_size"
                initialValue="(right - left) / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<Intrinsic> tag

```
<Type Name="Line">
    <Intrinsic Name="length" Expression="right - left"/>
    <DisplayString>{{ length={right - left} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">right - left</Item>
        <CustomListItems>
            <Variable Name="steps" initialValue="5"/>
            <Variable Name="step_size"
                initialValue="(right - left) / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<Intrinsic> tag

```
<Type Name="Line">
    <Intrinsic Name="length" Expression="right - left"/>
    <DisplayString>{{ length={length()} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">length()</Item>
        <CustomListItems>
            <Variable Name="steps" initialValue="5"/>
            <Variable Name="step_size"
                initialValue="length() / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



<Intrinsic> tag

```
<Type Name="Line">
    <Intrinsic Name="length" Expression="right - left"/>
    <DisplayString>{{ length={length()} }}</DisplayString>
    <Expand HideRawView="true">
        <Item Name="[length]">length()</Item>
        <CustomListItems>
            <Variable Name="steps" InitialValue="5"/>
            <Variable Name="step_size"
                initialValue="length() / (steps - 1)"/>
            <Variable Name="current" initialValue="left"/>
            <Variable Name="index" initialValue="0"/>
            ...
        </CustomListItems>
    </Expand>
</Type>
```



Official docs



Official docs

"Create custom views of C++ objects in the debugger using the Natvis framework"



Official docs

"Create custom views of C++ objects in the debugger using the Natvis framework"

- <https://learn.microsoft.com/en-us/visualstudio/debugger/create-custom-views-of-native-objects>



Official docs

"Create custom views of C++ objects in the debugger using the Natvis framework"

- <https://learn.microsoft.com/en-us/visualstudio/debugger/create-custom-views-of-native-objects>

"Implement NatVis custom intrinsic function for C++"



Official docs

"Create custom views of C++ objects in the debugger using the Natvis framework"

- <https://learn.microsoft.com/en-us/visualstudio/debugger/create-custom-views-of-native-objects>

"Implement NatVis custom intrinsic function for C++"

- <https://learn.microsoft.com/en-us/visualstudio/debugger/implementing-natvis-intrinsic-function>



Official docs

"Create custom views of C++ objects in the debugger using the Natvis framework"

- <https://learn.microsoft.com/en-us/visualstudio/debugger/create-custom-views-of-native-objects>

"Implement NatVis custom intrinsic function for C++"

- <https://learn.microsoft.com/en-us/visualstudio/debugger/implementing-natvis-intrinsic-function>
- New article from January 2025



My articles



My articles

"Natvis for boost::unordered_map, and how to use <Intrinsic> elements"



My articles

"Natvis for boost::unordered_map, and how to use <Intrinsic> elements"

- <https://blog.ganetsky.com/NatvisForUnordered/>



My articles

"Natvis for boost::unordered_map, and how to use <Intrinsic> elements"

- <https://blog.ganetsky.com/NatvisForUnordered/>

"Natvis for boost::concurrent_flat_map, and why fancy pointers are hard"



My articles

"**Natvis for boost::unordered_map, and how to use <Intrinsic> elements**"

- <https://blog.ganets.ky/NatvisForUnordered/>

"**Natvis for boost::concurrent_flat_map, and why fancy pointers are hard**"

- <https://blog.ganets.ky/NatvisForUnordered2/>



My articles

"**Natvis for boost::unordered_map, and how to use <Intrinsic> elements**"

- <https://blog.ganets.ky/NatvisForUnordered/>

"**Natvis for boost::concurrent_flat_map, and why fancy pointers are hard**"

- <https://blog.ganets.ky/NatvisForUnordered2/>

"**I want to write automated Natvis testing**"



My articles

"**Natvis for boost::unordered_map, and how to use <Intrinsic> elements**"

- <https://blog.ganets.ky/NatvisForUnordered/>

"**Natvis for boost::concurrent_flat_map, and why fancy pointers are hard**"

- <https://blog.ganets.ky/NatvisForUnordered2/>

"**I want to write automated Natvis testing**"

- <https://blog.ganets.ky/NatvisTesting/>



For a subsequent talk



For a subsequent talk

- **Optional attribute for SFINAE-like features**



For a subsequent talk

- Optional attribute for SFINAE-like features
- Navigating advanced `<Intrinsic>`



For a subsequent talk

- Optional attribute for SFINAE-like features
- Navigating advanced <Intrinsic>
- Writing Natvis to allow user-injected behaviour



For a subsequent talk

- **Optional attribute for SFINAE-like features**
- **Navigating advanced <Intrinsic>**
- **Writing Natvis to allow user-injected behaviour**
- **Attempts to automate testing of Natvis**



For a subsequent talk

- Optional attribute for SFINAE-like features
- Navigating advanced <Intrinsic>
- Writing Natvis to allow user-injected behaviour
- Attempts to automate testing of Natvis
- My Natvis wish list



GDB pretty-printer setup

GDB pretty-printer setup



GDB pretty-printer setup

- Basic GDB



GDB pretty-printer setup

- Basic GDB
- .gdbinit file



GDB pretty-printer setup

- Basic GDB
- .gdbinit file
- **Linking custom pretty-printers**



GDB pretty-printer setup

- Basic GDB
- .gdbinit file
- Linking custom pretty-printers
- Plumbing the pretty-printers



Basic GDB



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```



Basic GDB

```
> clang++ main.cpp
```

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp
```

```
> gdb
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Type "apropos word" to search for commands related to "word".
(gdb)
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Type "apropos word" to search for commands related to "word".
(gdb)
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Type "apropos word" to search for commands related to "word".
(gdb) file a.out
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Type "apropos word" to search for commands related to "word".
(gdb) file a.out
Reading symbols from a.out...
(No debugging symbols found in a.out)
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```



Basic GDB

```
> clang++ main.cpp -g
```

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp -g
```

```
> gdb
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
> clang++ main.cpp -g
> gdb
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Type "apropos word" to search for commands related to "word".
(gdb) file a.out
Reading symbols from a.out...
```



Basic GDB

```
(gdb) start
```

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
(gdb) print z
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
(gdb) print z
$1 = 0
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
(gdb) print z
$1 = 0
(gdb) next
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
(gdb) print z
$1 = 0
(gdb) next
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
(gdb) print z
$1 = 0
(gdb) next
5     }
(gdb) print z
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) start
Temporary breakpoint 1 at 0x1134: file main.cpp, line 2.
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Temporary breakpoint 1, main () at main.cpp:2
2         int x = 5;
(gdb) next
3         int y = 6;
(gdb) next
4         int z = x + y;
(gdb) print z
$1 = 0
(gdb) next
5     }
(gdb) print z
$2 = 11
```



Basic GDB

```
(gdb) file a.out
Reading symbols from a.out...
```

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) file a.out
Reading symbols from a.out...
(gdb) break 5
Breakpoint 1 at 0x114b: file main.cpp, line 5.
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) file a.out
Reading symbols from a.out...
(gdb) break 5
Breakpoint 1 at 0x114b: file main.cpp, line 5.
(gdb) run
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".

Breakpoint 1, main () at main.cpp:5
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) file a.out
Reading symbols from a.out...
(gdb) break 5
Breakpoint 1 at 0x114b: file main.cpp, line 5.
(gdb) run
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".

Breakpoint 1, main () at main.cpp:5
5 }
```



Basic GDB

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
(gdb) file a.out
Reading symbols from a.out...
(gdb) break 5
Breakpoint 1 at 0x114b: file main.cpp, line 5.
(gdb) run
Starting program: path/to/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".

Breakpoint 1, main () at main.cpp:5
5
(gdb) print z
$1 = 11
```



.gdbinit file

.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```

.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```

```
> gdb
```



.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
warning: File "path/to/.gdbinit" auto-loading has been
declined by your `auto-load safe-path' set to "...".
To enable execution of this file add
    add-auto-load-safe-path path/to/.gdbinit
line to your configuration file
"/home/braden/.config/gdb/gdbinit".
...etc...
```



.gdbinit file

```
# /home/braden/.config/gdb/gdbinit  
add-auto-load-safe-path path/to/.gdbinit
```



.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```



.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```

> gdb



.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```

> gdb

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Breakpoint 1 at 0x114b: file main.cpp, line 5.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Breakpoint 1, main () at main.cpp:5
5 }
```



.gdbinit file

```
0 // main.cpp
1 int main() {
2     int x = 5;
3     int y = 6;
4     int z = x + y;
5 }
```

```
# .gdbinit
file a.out
break 5
run
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Breakpoint 1 at 0x114b: file main.cpp, line 5.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "path/to/libthread_db.so.1".
```

```
Breakpoint 1, main () at main.cpp:5
5      }
(gdb) print z
$1 = 11
```



Linking custom pretty-printers

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 11 00 01 01 11 1



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

```
# .gdbinit
set print pretty on
source MyTypePrinter.py
file a.out
break 6
run
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

```
# .gdbinit
set print pretty on
source MyTypePrinter.py
file a.out
break 6
run
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

```
# .gdbinit
set print pretty on
source MyTypePrinter.py
file a.out
break 6
run
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

```
# .gdbinit
set print pretty on
source MyTypePrinter.py
file a.out
break 6
run
```

```
> gdb
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"
```

```
# ...etc for the plumbing...
```

```
# .gdbinit
set print pretty on
source MyTypePrinter.py
file a.out
break 6
run
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Breakpoint 1, main () at main.cpp:6
6 }
```



Linking custom pretty-printers

```
0 // main.cpp
1 class MyType {
2     // ...
3 };
4 int main() {
5     MyType my_type{};
6 }
```

```
# MyTypePrinter.py
class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

# ...etc for the plumbing...
```

```
# .gdbinit
set print pretty on
source MyTypePrinter.py
file a.out
break 6
run
```

```
> gdb
```

```
GNU gdb (GDB) 14.0.50.20221220-git
Copyright (C) 2022 Free Software Foundation, Inc.
...etc...
Breakpoint 1, main () at main.cpp:6
6
(gdb) print my_type
$1 = Hello, printer!
```



Plumbing the pretty-printers

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 11 00 01 01 11 1



Plumbing the pretty-printers

```
import gdb.printing
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"
```

Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
    return pp
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
    return pp
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
    return pp

gdb.printing.register.pretty_printer(
)
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
    return pp

gdb.printing.register.pretty_printer(
    gdb.current_objfile(),
    build_pretty_printer()
)
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
    return pp

gdb.printing.register.pretty_printer(
    gdb.current_objfile(),
    build_pretty_printer()
)
```



Plumbing the pretty-printers

```
import gdb.printing

class MyTypePrinter:
    def __init__(self, val):
        pass
    def to_string(self):
        return "Hello, printer!"

def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")
    pp.add_printer('MyType', '^MyType$', MyPrinter)
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)
    return pp

gdb.printing.register.pretty_printer(
    gdb.current_objfile(),
    build_pretty_printer()
)
```



Writing GDB pretty- printers

GDB pretty-printers



GDB pretty-printers

- Python



GDB pretty-printers

- Python
- Therefore, full power of a programming language



GDB pretty-printers

- Python
- Therefore, full power of a programming language
- E.g. external libraries

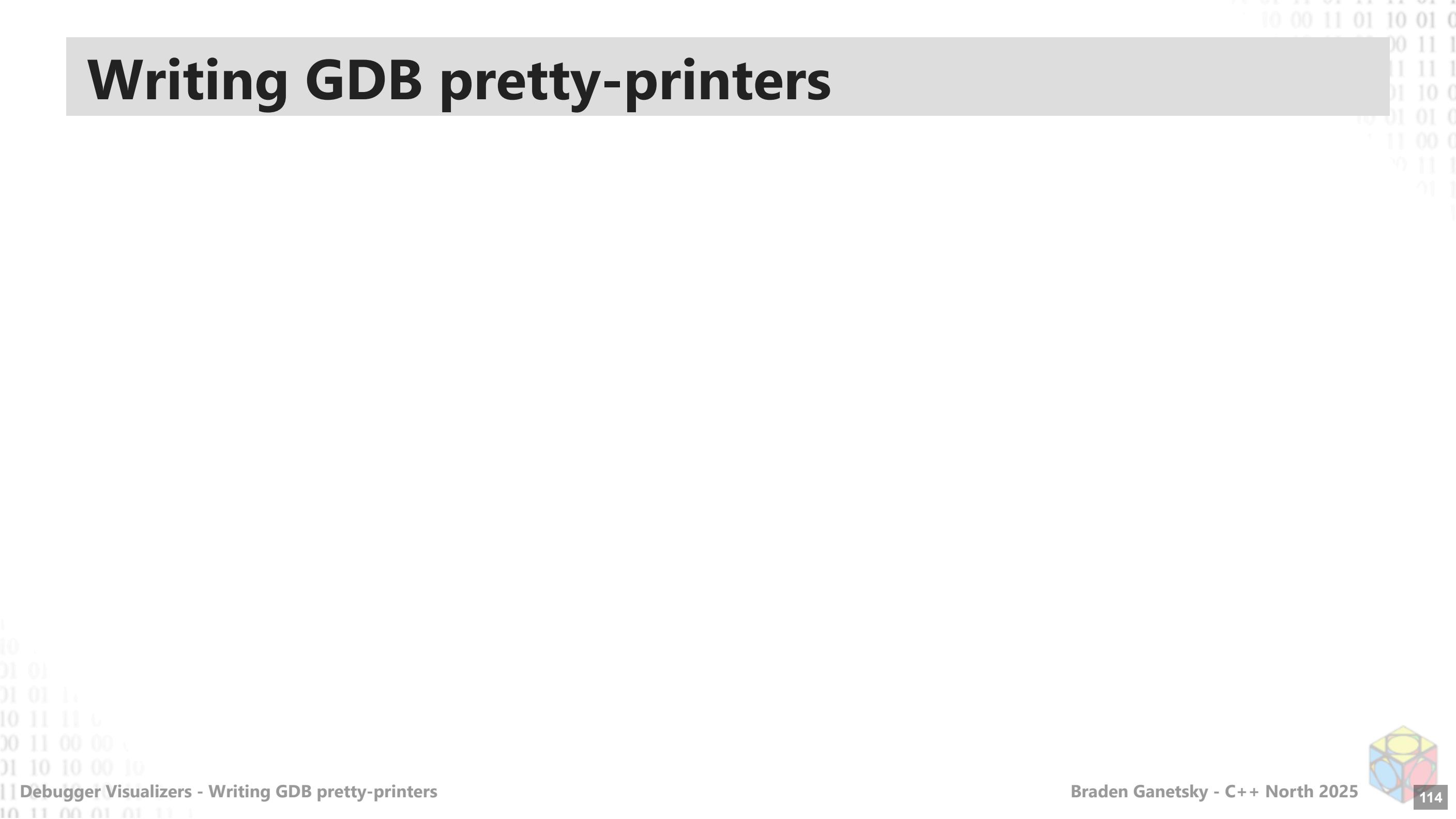


GDB pretty-printers

- Python
- Therefore, full power of a programming language
- E.g. external libraries
- **Easier to write than Natvis (in my opinion)**



Writing GDB pretty-printers



Writing GDB pretty-printers

- Constructor and `to_string()` method



Writing GDB pretty-printers

- Constructor and `to_string()` method
- `children()` method



Writing GDB pretty-printers

- Constructor and `to_string()` method
- `children()` method
- Using GDB from VSCode



Constructor and `to_string()`

```
class MyTypePrinter:  
    def __init__(self, val):  
        pass  
    def to_string(self):  
        return "Hello, printer!"
```

```
def build_pretty_printer():  
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")  
    pp.add_printer('MyType', '^MyType$', MyPrinter)  
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)  
    return pp
```

```
gdb.printing.register_pretty_printer(  
    gdb.current_objfile(),  
    build_pretty_printer()  
)
```



Constructor and `to_string()`

```
class MyTypePrinter:  
    def __init__(self, val):  
        pass  
    def to_string(self):  
        return "Hello, printer!"
```

```
def build_pretty_printer():  
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")  
    pp.add_printer('MyType', '^MyType$', MyPrinter)  
    pp.add_printer('MyTemplateType', '^MyTemplateType<.*>$', MyPrinter)  
    return pp
```

```
gdb.printing.register_pretty_printer(  
    gdb.current_objfile(),  
    build_pretty_printer()  
)
```

```
struct Line {  
    int left;  
    int right;  
};
```



Constructor and `to_string()`

```
class LinePrinter:  
    def __init__(self, val):  
        pass  
    def to_string(self):  
        return "Hello, printer!"
```

```
def build_pretty_printer():  
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_printers")  
    pp.add_printer('Line', '^Line$', LinePrinter)  
  
    return pp
```

```
gdb.printing.register_pretty_printer(  
    gdb.current_objfile(),  
    build_pretty_printer()  
)
```

```
struct Line {  
    int left;  
    int right;  
};
```



Constructor

```
class LinePrinter:  
    def __init__(self, val):  
        pass  
  
    def to_string(self):  
        return "Hello, printer!"
```



Constructor

```
class LinePrinter:  
    def __init__(self, val):  
        pass  
  
    def to_string(self):  
        return "Hello, printer!"
```



Constructor

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        return "Hello, printer!"
```



Constructor

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        return "Hello, printer!"
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        return "Hello, printer!"
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        return "Hello, printer!"
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        return f"{{{left}, {right}}}"
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        return f"{{{left}, {right}}}"
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        return f"{{{left}, {right}}}"
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        return f"{{{left}, {right}}}"
```

```
Line line{ 100, 600 };
```



to_string() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        return f"{{{left}, {right}}}"
```

```
Line line{ 100, 600 };
```

```
(gdb) print line  
$1 = {100, 600}
```



children() method

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 00 01 01 11 1



children() method

- Must return a Python iterator



children() method

- Must return a Python iterator
- Each item is a name-value pair

children() method

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 00 01 01 11 1



children() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        length = right - left  
        return f"Line of length {length}"
```



children() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        length = right - left  
        return f"Line of length {length}"
```



children() method

```
class LinePrinter:  
    def __init__(self, val):  
        self.val = val  
  
    def to_string(self):  
        left = self.val["left"]  
        right = self.val["right"]  
        length = right - left  
        return f"Line of length {length}"  
  
    def children(self):  
        def generator():  
            # ...  
        return generator()
```



children() method

```
def children(self):
    def generator():

        return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        while current < right:
            yield current
            current += step_size
            index += 1

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:

            return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current
            index += 1
            current = left + index * step_size

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current
            index += 1
            current = left + index * step_size

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current
            index += 1
            current = left + index * step_size

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current
            index += 1
            current = left + index * step_size

    return generator()
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current
            index += 1
            current = left + index * step_size

    return generator()
```

```
Line line{ 100, 600 };
```



children() method

```
def children(self):
    def generator():
        left = self.val["left"]
        right = self.val["right"]

        steps = 5
        step_size = (right - left) / (steps - 1)
        current = left
        index = 0

        yield "steps", steps

        while current <= right:
            yield f"[{index}]", current
            index += 1
            current = left + index * step_size

    return generator()
```

```
Line line{ 100, 600 };
```

```
(gdb) print line
$1 = Line of length 500 = {
    steps = 5,
    [0] = 100,
    [1] = 225,
    [2] = 350,
    [3] = 475,
    [4] = 600
}
```



Using GDB from VSCode



Using GDB from VSCode

- Take advantage of VSCode's GUI



Using GDB from VSCode

- Take advantage of VSCode's GUI
- Cannot use ".gdbinit" file



Using GDB from VSCode

- Take advantage of VSCode's GUI
- Cannot use ".gdbinit" file
- Requires GDB 14.1 otherwise `to_string()` is broken



Using GDB from VSCode

- Take advantage of VSCode's GUI
- Cannot use ".gdbinit" file
- Requires GDB 14.1 otherwise `to_string()` is broken
- (https://sourceware.org/bugzilla/show_bug.cgi?id=11335)



Using GDB from VSCode

1
0
01 01
01 01 11
10 11 11 0
00 11 00 00
01 10 10 00 10
11 11 00 01 01 11 1



Using GDB from VSCode

```
// .vscode/launch.json
{
    "version": "0.2.0",
    "configurations": [
        {
            // ...
        }
    ]
}
```

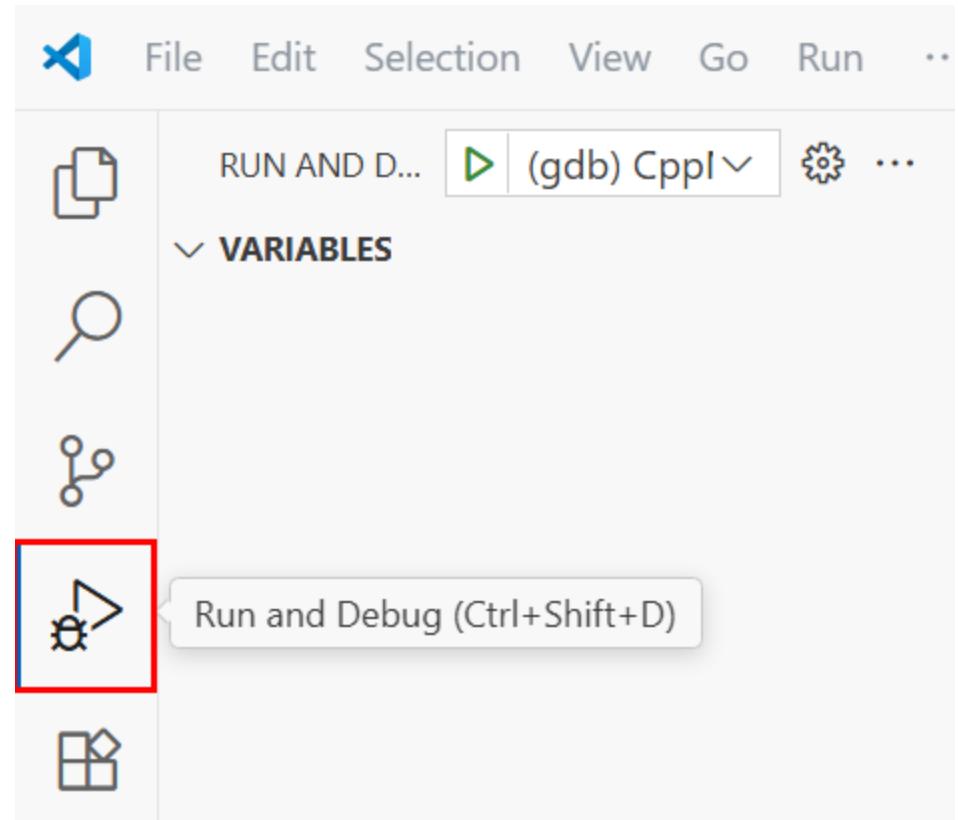


Using GDB from VSCode

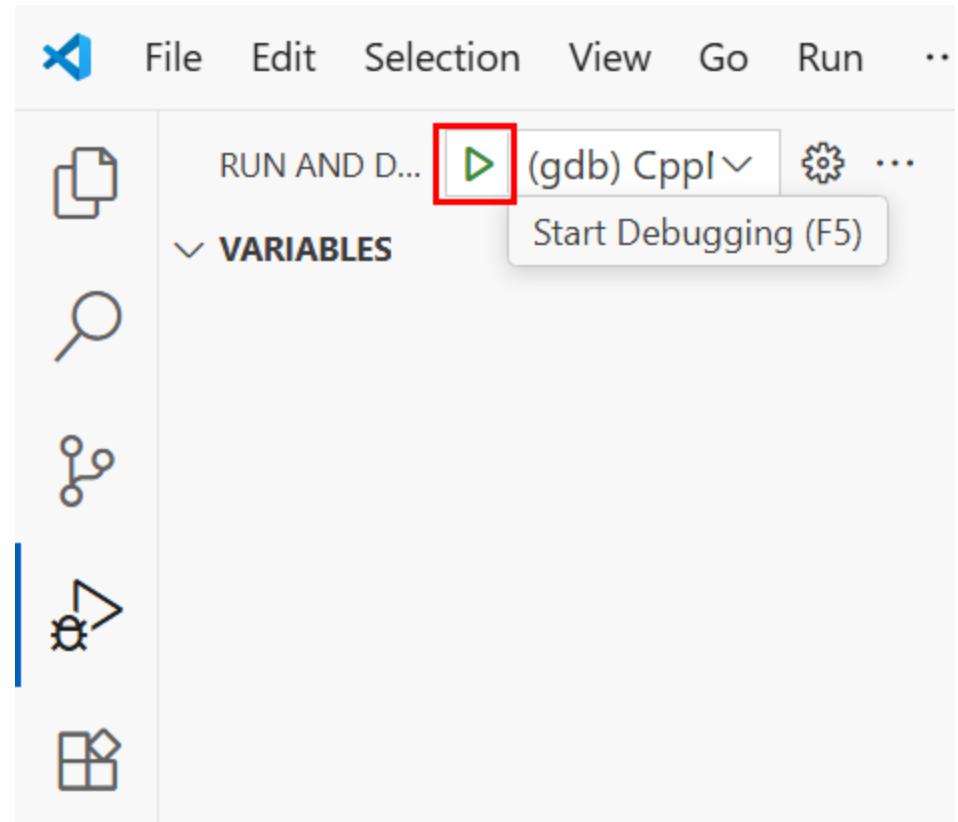
```
// .vscode/launch.json
{
    "version": "0.2.0",
    "configurations": [
        {
            // ...
            "setupCommands": [
                // ...
            ]
        }
    ]
}
```



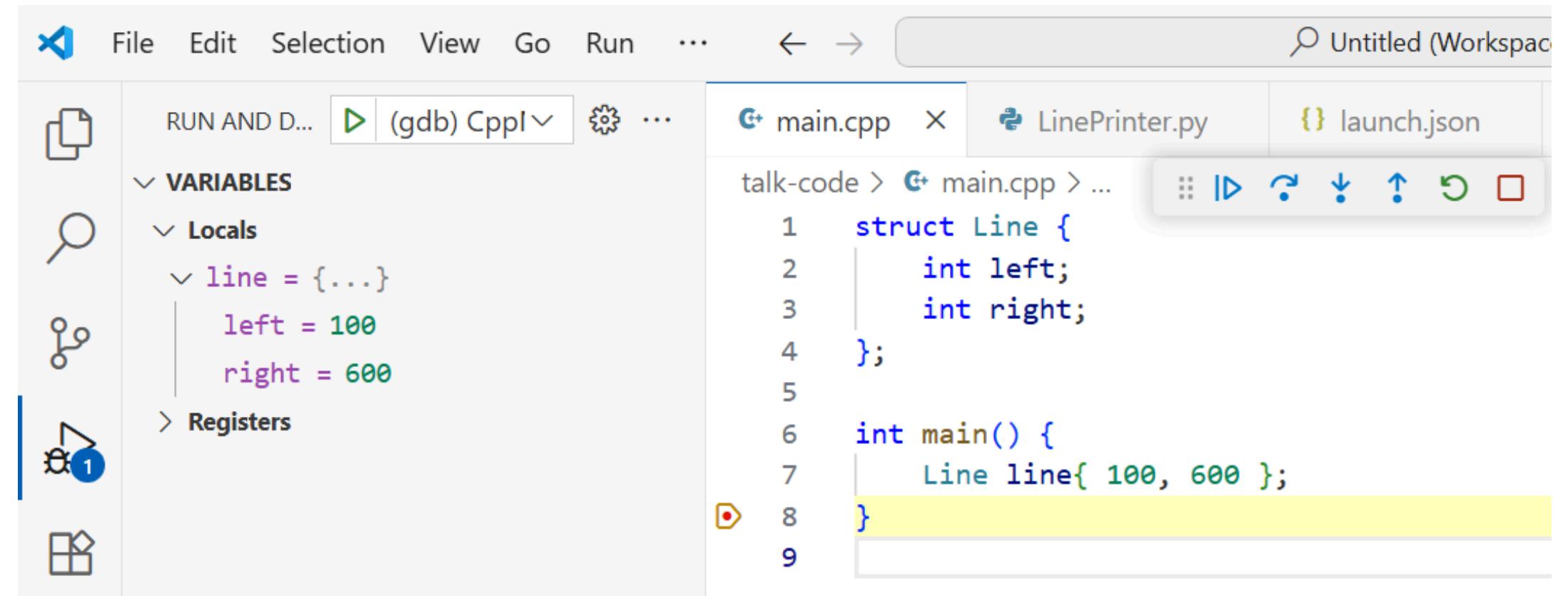
Using GDB from VSCode



Using GDB from VSCode



Using GDB from VSCode



Using GDB from VSCode

```
"setupCommands": [
```

```
]
```



Using GDB from VSCode

```
"setupCommands": [  
    {  
        "description": "Enable pretty-printing for gdb",  
        "text": "-enable-pretty-printing",  
        "ignoreFailures": true  
    },  
]
```



Using GDB from VSCode

```
"setupCommands": [
    {
        "description": "Enable pretty-printing for gdb",
        "text": "-enable-pretty-printing",
        "ignoreFailures": true
    },
    {
        "description": "Load pretty printers",
        "text": "source ${workspaceFolder}/LinePrinter.py",
        "ignoreFailures": true
    }
]
```

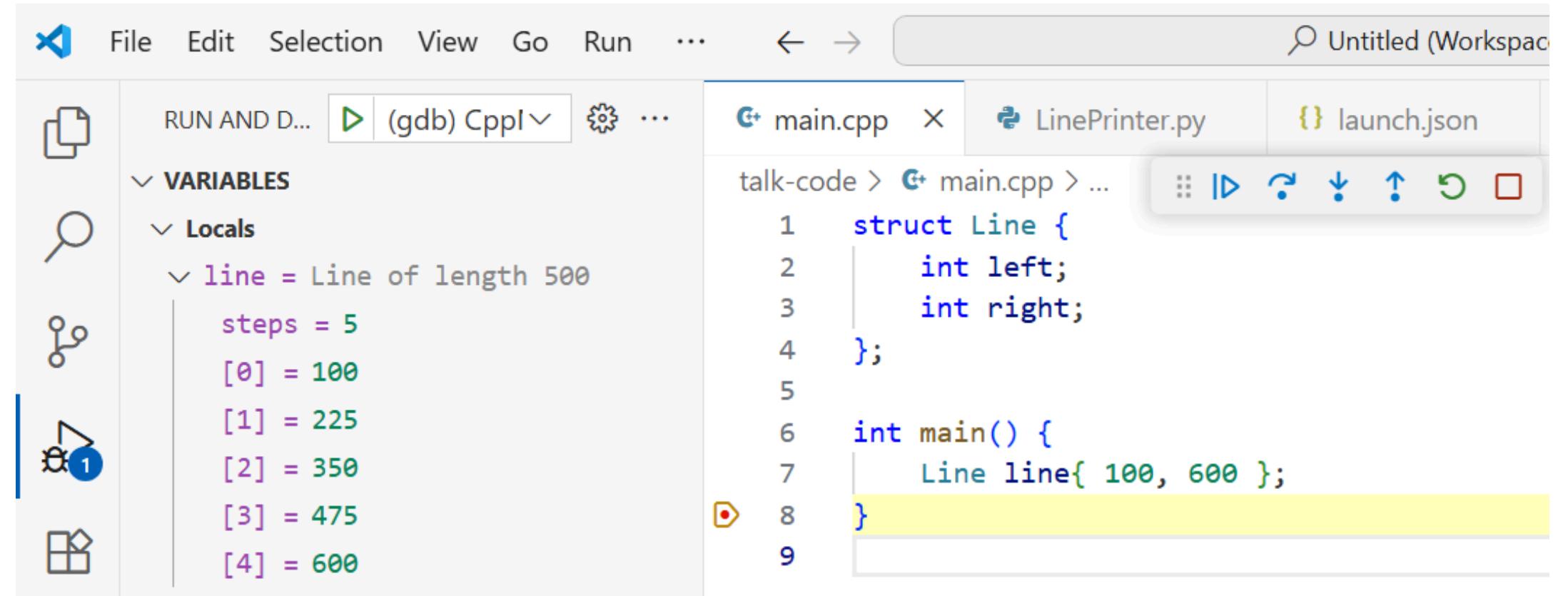


Using GDB from VSCode

```
"setupCommands": [
    {
        "description": "Enable pretty-printing for gdb",
        "text": "-enable-pretty-printing",
        "ignoreFailures": true
    },
    {
        "description": "Load pretty printers",
        "text": "source ${workspaceFolder}/LinePrinter.py",
        "ignoreFailures": true
    }
]
```



Using GDB from VSCode



The screenshot shows the Visual Studio Code (VSCode) interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Untitled (Workspace)
- Left Sidebar:** RUN AND D... (gdb) Cppl, VARIABLES, Locals, line = Line of length 500, steps = 5, [0] = 100, [1] = 225, [2] = 350, [3] = 475, [4] = 600.
- Central Area:** main.cpp (active), LinePrinter.py, launch.json.
- Code Editor:**谈-code > main.cpp > ...

```
1 struct Line {  
2     int left;  
3     int right;  
4 };  
5  
6 int main() {  
7     Line line{ 100, 600 };  
8 }  
9
```
- Bottom Status Bar:** Shows the current line number (8) and the status bar icon.



Resources



Resources

"Python API"



Resources

"Python API"

- <https://sourceware.org/gdb/current/onlinedocs/gdb.html/Python-API.html>



Resources

"Python API"

- <https://sourceware.org/gdb/current/onlinedocs/gdb.html/Python-API.html>

"Visualizing boost::unordered_map in GDB, with pretty-printer customization points"



Resources

"Python API"

- <https://sourceware.org/gdb/current/onlinedocs/gdb.html/Python-API.html>

"Visualizing boost::unordered_map in GDB, with pretty-printer customization points"

- <https://blog.ganetsky.com/PrettyPrinter/>



For a subsequent talk



For a subsequent talk

- Go into `gdb.Value` and `gdb.Type`



For a subsequent talk

- Go into `gdb.Value` and `gdb.Type`
- **Writing pretty-printers to allow user-injected behaviour**



For a subsequent talk

- Go into `gdb.Value` and `gdb.Type`
- Writing pretty-printers to allow user-injected behaviour
- `Xmethods`



For a subsequent talk

- Go into `gdb.Value` and `gdb.Type`
- Writing pretty-printers to allow user-injected behaviour
- Xmethods
- Embedding pretty-printers in the binary itself



Conclusion

Write visualizers



Write visualizers

- Now you know how!



Write visualizers

- Now you know how!
- Writing a visualizer is accessible to you!



Write visualizers

- Now you know how!
- Writing a visualizer is accessible to you!
- Colleagues will thank you



Write visualizers

- Now you know how!
- Writing a visualizer is accessible to you!
- Colleagues will thank you
- Your future self will thank you



Ultimately I want



Ultimately I want

- **Developer empowerment**



Ultimately I want

- **Developer empowerment**
- **Removal of barriers to understanding**



Ultimately I want

- **Developer empowerment**
- **Removal of barriers to understanding**
- **Libraries that are more accessible to use**



Ultimately I want

- **Developer empowerment**
- **Removal of barriers to understanding**
- **Libraries that are more accessible to use**
- **Widely consumed libraries to have debugger support**



Our work is about more than the source code



Our work is about more than the source code

Our work is about people

Our work is about more than the source code

Our work is about people

Let's make our code more accessible

Thank you!

Braden Ganetsky

braden@ganets.ky
GitHub @k3DW

