Bloomberg
Engineering

undo

mosaic
CONSULTANTS TO FINANCIAL SERVICES

Generic Programming Without (Writing Your Own) Templates

Tina Ulbrich

Templates

Templates

```cpp
#include <iostream>

template <typename T>
void print(const T& val)
{
    std::cout << val << "\n";
}

int main()
{
    print(3);
    print(2.0);
}
```

```
3
2
```

Templates

```cpp
#include <iostream>

template <typename T>
void print(const T& val)
{
    std::cout << val << "\n";
}

struct date
{
    int day;
    int month;
    int year;
};

int main()
{
    const auto accu_talk_date = date{ .day = 11, .month = 3, .year = 2021 };
    print(accu_talk_date);
}
```

Templates

```
Build started...
1>------ Build started: Project: highlighting, Configuration: Debug Win32 ------
1>highlighting.cpp
1>D:\programs\accu2021\examples.cpp(112,1): error C2679: binary '<<': no operator found which takes a right-hand operand of type 'const T' (or there is no acceptable conversion)
1>        with
1>        [
1>            T=date
1>        ]
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(465,39): message : could be 'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(std::basic_streambuf<char,std::char_trait
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(440,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(const void *)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(422,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(long double)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(404,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(double)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(386,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(float)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(368,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(unsigned __int64)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(350,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(__int64)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(332,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(unsigned long)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(314,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(long)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(295,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(unsigned int)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(269,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(int)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(250,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(unsigned short)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(216,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(short)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(198,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(bool)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(193,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(std::ios_base &(__cdecl *)(std::ios_base
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(188,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(std::basic_ios<char,std::char_traits<char
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(183,39): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(std::basic_ostream<char,std::char_traits<
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(460,20): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<<void>(std::nullptr_t)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(653,32): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<char,std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(697,32): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<char,std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(733,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const char
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(778,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(895,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const signe
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(901,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,signed char
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(906,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const unsig
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(912,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,unsigned ch
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(920,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const char8
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(926,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char8_t)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(934,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,wchar_t)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(937,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const wchar
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(941,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char16_t)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(943,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char32_t)'
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(951,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const char1
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(953,31): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const char3
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333\include\ostream(998,32): message : or       'std::basic_ostream<char,std::char_traits<char>> &std::operator <<<char,std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const
1>D:\programs\accu2021\examples.cpp(112,1): message : while trying to match the argument list '(std::ostream, const T)'
1>        with
1>        [
1>            T=date
1>        ]
1>D:\programs\accu2021\examples.cpp(127): message : see reference to function template instantiation 'void print<date>(const T &)' being compiled
1>        with
1>        [
1>            T=date
1>        ]
1>Done building project "highlighting.vcxproj" -- FAILED.
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

Templates

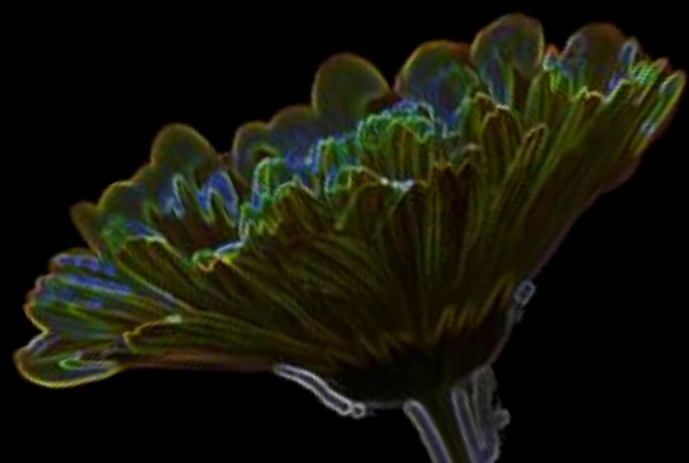
```
1>D:\programs\accu2021\examples.cpp(112,1):
    error C2679: binary '<<': no operator found which takes a right-hand operand
    of type 'const T‘ (or there is no acceptable conversion)
1>          with
1>          [
1>              T=date
1>          ]
```

Templates

```
template <typename T>
concept streamable = requires (std::ostream s, T val) { s << val; };

template <typename T>
void print(const T& val) requires streamable<T>
{
    std::cout << val << "\n";
}
```

```
1>D:\programs\accu2021\examples.cpp(131,5): error C2672: 'print': no matching overloaded function
found

1>D:\programs\accu2021\examples.cpp(131,25): error C7602: 'print': the associated constraints are
not satisfied

1>D:\programs\accu2021\examples.cpp(114): message : see declaration of 'print'
```

std::span



Why span?

```cpp
auto mean(const std::vector<double>& range)
{
    return std::accumulate(range.begin(), range.end(), 0.0) / static_cast<double>(range.size());
}
```

~~~~cpp
auto mean(const std::array<double>& range)
{
    return std::accumulate(range.begin(), range.end(), 0.0) / static_cast<double>(range.size());
}
~~~~

```cpp
template <typename T>
auto mean(const T& range)
{
    return std::accumulate(range.begin(), range.end(), 0.0) / static_cast<double>(range.size());
}
```

Why span?

```
auto mean(const std::span<const double> range)
{
    return std::accumulate(range.begin(), range.end(), 0.0) / static_cast<double>(range.size());
}
```

```
const auto v = std::vector(10, 0.0);
const auto mean_v = mean(v);
```

```
const auto a = std::array{ 1.0, 2.0, 3.0 };
const auto mean_a = mean(a);
```

```
const double a[] = { 1.0, 2.0, 3.0, 4.0, 5.0 };
const auto mean_a = mean(a);
```

How to use a span

```
const auto vec = std::vector({ 1.0, 3.0, 4.0, 7.0, 4.0, 7.0, 2.0, 5.0 });
std::cout << mean(vec) << std::endl;
```

```
std::cout << mean({ &vec[2], &vec[vec.size() - 2] }) << std::endl;
```

```
std::cout << mean({ vec.begin() + 2, vec.end() - 2 }) << std::endl;
```

```
std::cout << mean(vec | drop(2) | drop_last(2)) << std::endl;
```

```
const auto s = std::span(vec) | stride(2);
```

Pitfalls

```
auto vec = std::vector(10, 0);
auto s = std::span(vec);
s[0] = 100;
```

```
vec { 100, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
```

Pitfalls

```
auto vec = std::vector(10, 0);
auto s = std::span(vec);
s[0] = 100;
```

```
const auto vec = std::vector(10, 0);
auto s = std::span(vec);
s[0] = 100;
```

```
const auto vec = std::vector(10, 0);
const auto s = std::span(vec);
s[0] = 100;
```

```
auto vec = std::vector(10, 0);
const auto s = std::span(vec);
s[0] = 100;
```

```
vec { 100, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
```

Pitfalls

```
auto vec = std::vector(100, 0);
const auto s = std::span<const int>(vec);
s[0] = 100;
```

Pitfalls

```
double& return_a_ref()
{
    const auto value = 0.0;
    const auto& zero = value;
    return zero;
}

const auto zero = return_a_ref();
```

```
auto return_a_span()
{
    const auto vec = std::vector(10, 10);
    const auto s = std::span(vec);
    return s;
}

const auto s = return_a_span()
```

Pitfalls

```
auto make_span(const boost::multi_array<double, 1>& multi_array)
{
    return std::span(multi_array.origin(), multi_array.num_elements());
}

const auto arr = boost::multi_array<double, 1>(boost::extens[10]);
const auto sp = make_span(arr);
```

Pitfalls

```
auto a_vector()
{
    return std::vector(10, 5);
}

const auto vec1 = a_vector();
const auto& vec2 = a_vector();
const auto s = std::span<const int>(a_vector());
```

No C++20?

```
auto mean(const std::span<const double> range)
{
    return std::accumulate(range.begin(), range.end(), 0.0) / static_cast<double>(range.size());
}
```

```
const auto v = std::vector(10, 0.0);
const auto mean_v = mean(v);
```

```
const auto a = std::array{ 1.0, 2.0, 3.0 };
const auto mean_a = mean(a);
```

```
const double a[] = { 1.0, 2.0, 3.0, 4.0, 5.0 };
const auto mean_a = mean(a);
```

No C++20?

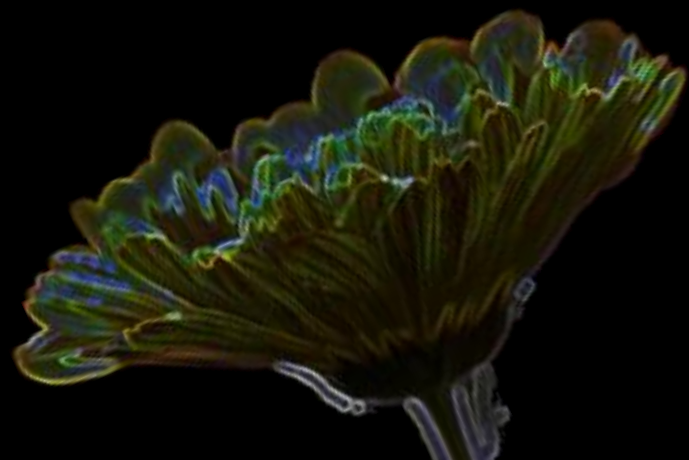
```cpp
auto mean(const gsl::span<const double> range)
{
    return std::accumulate(range.begin(), range.end(), 0.0) / static_cast<double>(range.size());
}
```

```cpp
const auto v = std::vector(10, 0.0);
const auto mean_v = mean(v);
```

```cpp
const auto a = std::array{ 1.0, 2.0, 3.0 };
const auto mean_a = mean(a);
```

```cpp
const double a[] = { 1.0, 2.0, 3.0, 4.0, 5.0 };
const auto mean_a = mean(a);
```

Questions?



std::variant

a type-safe union?

a type-safe union?

```
union types
{
    short  short_val;
    int    int_val;
    double double_val;
};
```

a type-safe union?

```
union types
{
    short  short_val;
    int    int_val;
    double double_val;
};
```

```
int main()
{
    auto t = types();

    t.int_val = 3;
    std::cout << t.int_val << "\n";

    t.double_val = 5.0;
    std::cout << t.double_val << "\n";

    std::cout << t.int_val << "\n";
}
```

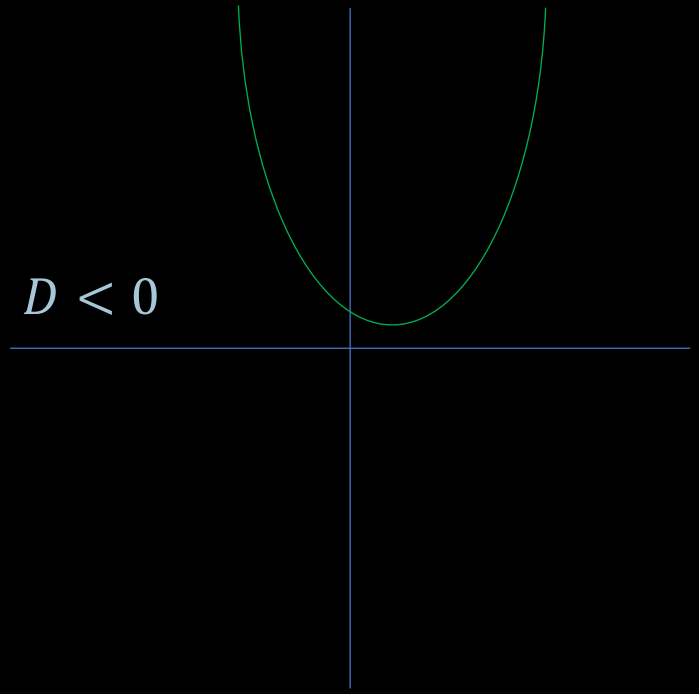
undefined behavior

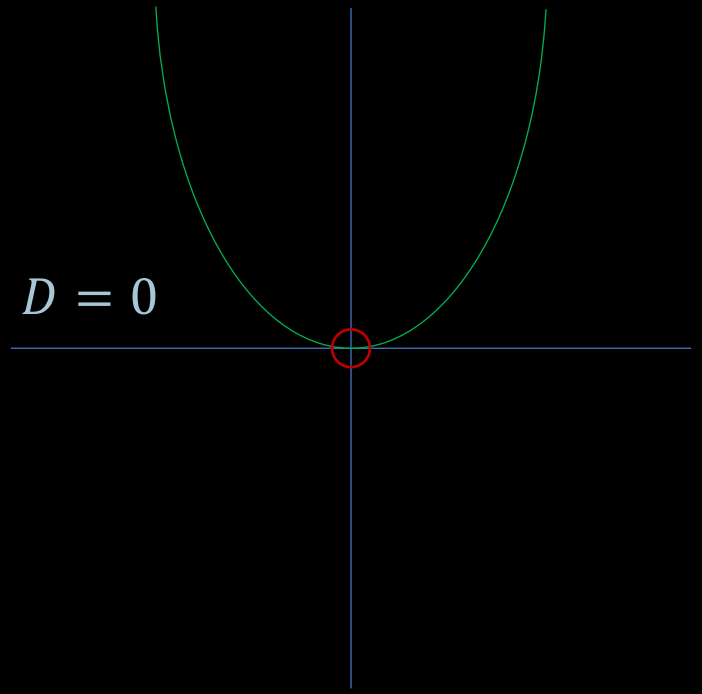
Example

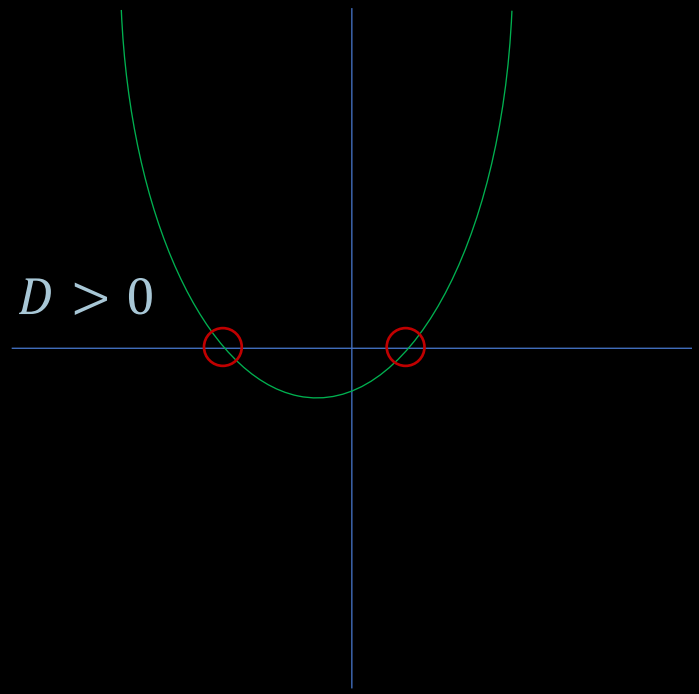
$$y = ax^2 + bx + c$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$D = b^2 - 4ac$$







Example

```
using roots = std::variant<std::pair<double, double>, double, std::monostate>;
```

```
roots calculate_roots(const double a, const double b, const double c)
{
    const auto d = b * b - 4 * a * c;

    if (d > 0.0)
    {
        const auto p = std::sqrt(d) / (2 * a);
        return std::pair{ -b + p, -b - p };
    }

    if (d == 0.0)
    {
        return -b / (2 * a);
    }

    return std::monostate();
}
```

Example

```cpp
const auto result = calculate_roots(10.0, -2.0, -5.0);

if (std::holds_alternative<std::pair<double, double>>(result))
{
    const auto [first_result, second_result] = std::get<std::pair<double, double>>(result);
    std::cout << "2 roots: " << first_result << " " << second_result << "\n";
}

if (std::holds_alternative<double>(result))
{
    std::cout << "1 root: " << std::get<double>(result) << "\n";
}

if (std::holds_alternative<std::monostate>(result))
{
    std::cout << "No roots found.\n";
}
```

Example

```cpp
const auto index = result.index();

if (index == 0)
{
    const auto [first_result, second_result] = std::get<0>(result);
    std::cout << "2 roots: " << first_result << " " << second_result << "\n";
}

if (index == 1)
{
    std::cout << "1 root: " << std::get<1>(result) << "\n";
}

if (index == 2)
{
    std::cout << "No roots found.\n";
}
```

visit

```
struct root_printer
{
    void operator()(const std::pair<double, double>& arg) const
    {
        std::cout << "2 roots: " << arg.first << " " << arg.second << '\n‘;
    }

    void operator()(const double arg) const
    {
        std::cout << "1 root: " << arg << '\n‘;
    }

    void operator()(const std::monostate) const
    {
        std::cout << "No roots found.\n";
    }
};
```

visit

```
int main()
{
    const auto printer = root_printer();
    std::visit(printer, calculate_roots(10, 0, -2));
    std::visit(printer, calculate_roots(2, 1, 2));
}
```

```
2 roots: 0.447214 -0.447214
No roots found.
```

```
int main()
{
    std::visit([](const auto& element) { std::cout << element; }, calculate_roots(10, 0, -2));
}
```

overload pattern

```
template<class... Ts> struct overload : Ts... { using Ts::operator()...; };
template<class... Ts> overload(Ts...)->overload<Ts...>;
```

```
std::visit(overload{
    [](const std::pair<double, double>& arg)
    {
        std::cout << "2 roots: " << arg.first << " " << arg.second << '\n‘;
    },

    [](const double arg)
    {
        std::cout << "1 root: " << arg << '\n‘;
    },

    [](const std::monostate)
    {
        std::cout << "No roots found.\n";
    }
}, calculate_roots(10.0, -2.0, -5.0));
```

Questions?

std::any

void*

```
val = 3
val = [1.1, 1.2, 1.3]
val = "Hello ACCU"
val = np.array([0, 1, 1, 2, 3, 5, 8, 13])
```

```
auto my_string = "Hello ACCU"s;
auto my_double = 2.0;
auto my_int = 1;

void* my_any = &my_string;
my_any = &my_double;
my_any = &my_int;

std::cout << *static_cast<int*>(my_any) << '\n';
std::cout << *static_cast<double*>(my_any) << '\n';
```

any

```
std::any my_any = "Hello ACCU"s;
my_any = 2.0;
my_any = 1;

std::cout << my_any.has_value() << '\n';
std::cout << my_any.type().name() << '\n';

std::cout << std::any_cast<int>(my_any) << '\n';
std::cout << std::any_cast<double>(my_any) << '\n';

my_any.reset();
my_any.emplace<my_complex_type>(my_complex_type{});
```

bad_any_cast

Use cases

```
struct property
{
    property() = default;
    property(const std::string_view name, const std::any& value) : name(name), value(value) {}

    std::string name;
    std::any value;
};

using properties = std::vector<property>;
```

No C++17?

Feature	Boost.Any	std::any
Extra memory allocation	Yes	Yes
Small buffer optimization	No	Yes
emplace	No	Yes
in_place_type_t in constructor	No	Yes

Further reading

How std::any Works

Published February 5, 2021

In the previous post we've seen a very nice technique to use value semantics with inheritance and virtual methods, which was made possible by `std::any`.

Inheritance Without Pointers

Published January 29, 2021

Inheritance is a useful but controversial technique in C++. There is even a famous talk by Sean Parent called Inheritance is the base class of evil. So inheritance is not the most popular feature of the C++ community.

Questions?

std::function

Example

```
template <typename Func>
auto filter_and_do_stuff(const std::vector<double>& data, const Func& condition)
{
    for (const auto element : data)
    {
        if (condition(element))
        {
            std::cout << element << "\n";
            // do stuff
        }
    }
}
```

```
int main()
{
    const auto data = std::vector{ 1.0, 2.0, 4.0, 0.0, 5.0 };
    filter_and_do_stuff(data, [](const double element) {return element > 2.0; });
    filter_and_do_stuff(data, [](const double element) {return element; });
}
```

Example

```cpp
auto filter_and_do_stuff(const std::vector<double>& data,
                         const std::function<bool(double)>& condition)
{
    for (const auto element : data)
    {
        if (condition(element))
        {
            std::cout << element << "\n";
            // do stuff
        }
    }
}
```

```cpp
int main()
{
    const auto data = std::vector{ 1.0, 2.0, 4.0, 0.0, 5.0 };
    filter_and_do_stuff(data, [](const double element) {return element > 2.0; });
    filter_and_do_stuff(data, [](const double element) {return element; });
}
```

function_ref

```
auto filter_and_do_stuff(const std::vector<double>& data,
                         const tl::function_ref<bool(double)>& condition)
{
    for (const auto element : data)
    {
        if (condition(element))
        {
            std::cout << element << "\n";
            // do stuff
        }
    }
}
```

```
int main()
{
    const auto data = std::vector{ 1.0, 2.0, 4.0, 0.0, 5.0 };
    filter_and_do_stuff(data, [](const double element) {return element > 2.0; });
    filter_and_do_stuff(data, [](const double element) {return element; });
}
```

Summary

Questions?

Tina Ulbrich (she/her) - @_Yulivee_ - ROSEN Technology and Research GmbH

ACCU 2021

Resources

general

- https://en.cppreference.com/
- https://cppinsights.io/

variant

- https://www.walletfox.com/course/patternmatchingcpp17.php
- https://www.bfilipek.com/2018/06/variant.html
- https://www.bfilipek.com/2019/02/2lines3featuresoverload.html
- https://docs.microsoft.com/en-gb/cpp/cpp/unions?view=msvc-160
- https://arne-mertz.de/2018/05/modern-c-features-stdvariant-and-stdvisit/

Resources

any

- https://www.bfilipek.com/2018/06/any.html
- https://dzone.com/articles/everything-you-need-to-know-about-stdany-from-c17
- https://www.geeksforgeeks.org/void-pointer-c-cpp/
- https://devblogs.microsoft.com/cppblog/stdany-how-when-and-why/
- https://www.nextptr.com/tutorial/ta157116485 12/stdany-comparison-with-void-and-motivating-examples
- https://www.geeksforgeeks.org/stdany-class-in-c/
- https://www.boost.org/doc/libs/1_75_0/doc/html/any.html
- https://www.fluentcpp.com/2021/02/05/how-stdany-works/
- https://www.fluentcpp.com/2021/01/29/inheritance-without-pointers/

Resources

function

- https://github.com/TartanLlama/function_ref
- http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0792r2.html
- https://blog.demofox.org/2015/02/25/avoiding-the-performance-hazzards-of-stdfunction/

gsl

- https://github.com/isocpp/CppCoreGuidelines/blob/master/docs/gsl-intro.md