ACCU Autumn 2019

The Secret Life of Numbers

John McFarlane

johnmcfarlane.github.io/slides/2019-accu

About Me

Software Engineer, Jaguar Land Rover, Shannon, Ireland

BSI, MISRA C++, WG21 (SG14 & SG6)

Background

BBC BASIC, 6502, C, 80286/7, C++, Python

Games, Fractals, AI, Simulation, APIs, Numerics, CI

Contents

Contents

Contents

Contents

Contents

1. Bit Parts

1.0. Acting Natural

1.0. Acting Natural

Question:

1.0. Acting Natural

Question: how many *digits* in `uint8_t`?

00101010

00101010 =

$$00101010 = 32 + 8 + 2$$

00101010 = 42

00101010

…0000000000000000101010

…0000000000000**00101010**.0000000000000…

…0000000000000000101010.0000000000000000…

Question: how many *digits* in `uint8_t`?

Question: how many *digits* in `uint8_t`?

Answer:

Question: how many *digits* in `uint8_t`?

Answer: ∞

Question: how many *digits* in `uint8_t`?

Answer: ∞+8

Question: how many *digits* in `uint8_t`?

Answer: ∞+8+∞

Question: how many *digits* in `uint8_t`?

Answer: ∞+8+∞ = 2∞+8

1.1. Be Positive

1.1. Be Positive

Question:

1.1. Be Positive

Question: how many *sign bits* in `int8_t`?

11010110

$$11010110 =$$

11010110 = -128+64+16+4+2

11010110 = -42

11010110

…1111111111111010110

…1111111111110101 10.0000000000000000…

…1111111111110101100.0000000000000000…

Question: how many *sign bits* in `int8_t`?

Question: how many *sign bits* in `int8_t`?

Answer:

Question: how many *sign bits* in `int8_t`?

Answer: ∞

Question: how many *sign bits* in `int8_t`?

Answer: ∞+1

1.2. A Slice of Π

1.2. A Slice of Π

Question:

1.2. A Slice of π

Question: how many *significant digits* in π?

11.0010010000111111011010101000…

011.001001000011111101101010100…

…0000011.00100100001111110110101010100…

Question: how many *significant digits* in π?

Question: how many *significant digits* in π?

Answer:

Question: how many *significant digits* in π?

Answer: 1

Question: how many *significant digits* in π?

Answer: 1 +2

Question: how many *significant digits* in π?

Answer: 1 +2 +∞

Question: how many *significant digits* in π?
Answer: 1 +2 +∞ =∞+3

1.3. Floatation Devices

1.3. Floatation Devices

Question:

1.3. Floatation Devices

Question: how many *digits* in `float`?

11.0010010000111111011010101000…

…0000011.00100100001111110110101010100…

…0000011.00100100001111110110101010100…

Question: how many *digits* in `float`?

Question: how many *digits* in `float`?

Answer:

Question: how many *digits* in `float`?

Answer: 24

Question: how many *digits* in `float`?

Answer: 24-1

Question: how many *digits* in `float`?
Answer: 24-1 = 23

1.4. Frame the Problem

1.4. Frame the Problem

Question:

1.4. Frame the Problem

Question: how many times does 42 go into 8?

…00000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000…

…0000000000101010.000000… 

Question: how many times does 42 go into 8?

Question: how many times does 42 go into 8?
Answer:

Question: how many times does 42 go into 8?
Answer: 3

1.9.9 Warning

Slide Code Ahead!

```
git clone https://github.com/johnmcfarlane/cnl.git
c++ -std=c++17 -Wall -Wextra -Werror -Wpedantic slide_code.cpp
```

```
#include <cnl/all.h>
#include <cnl/_impl/type_traits/identical.h>
using namespace cnl;
using namespace cnl::literals;
using namespace cnl::math;
using namespace std;
```

2. Typology

<0,0>

2.0. Don't Sell Yourself Short

2.0. Don't Sell Yourself Short

Question:

2.0. Don't Sell Yourself Short

Question: how do you *write* 'forty-two'?

CNL: Compositional Numeric Library

CNL: Compositional Numeric Library

github.com/johnmcfarlane/cnl

CNL: Compositional Numeric Library

github.com/johnmcfarlane/cnl

```
template<int Exponent=0, int Radix=2>
class power;
```

CNL: Compositional Numeric Library

github.com/johnmcfarlane/cnl

```
template<int Exponent=0, int Radix=2>
class power;
```

```
template<typename Rep=int, typename Scale=power<>>
class scaled_integer;
```

CNL: Compositional Numeric Library

github.com/johnmcfarlane/cnl

```
template<int Exponent=0, int Radix=2>
class power;
```

```
template<typename Rep=int, typename Scale=power<>>
class scaled_integer;
```

```
template<typename Rep, int Exponent, int Radix>
class scaled_integer<Rep, power<Exponent, Radix>> {
  // ...
private:
  Rep n;
};
```

CNL: Compositional Numeric Library

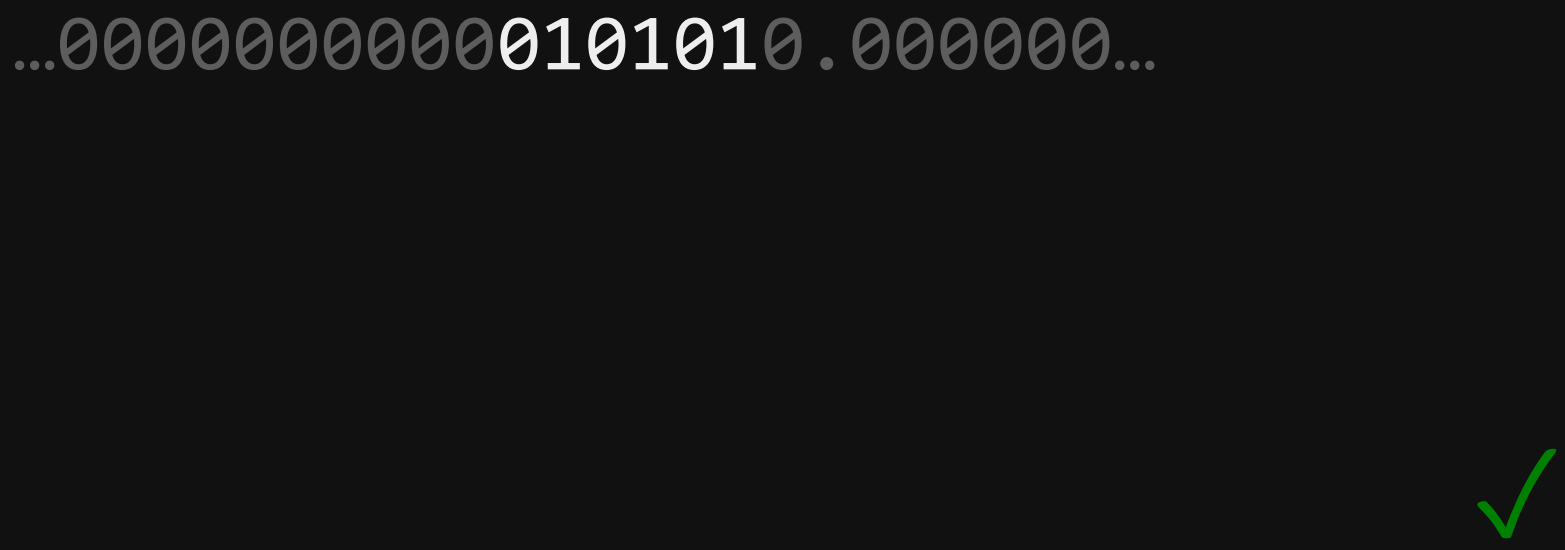
github.com/johnmcfarlane/cnl

```
template<int Exponent=0, int Radix=2>
class power;
```

```
template<typename Rep=int, typename Scale=power<>>
class scaled_integer;
```

```
template<typename Rep, int Exponent, int Radix>
class scaled_integer<Rep, power<Exponent, Radix>> {
  // ...
private:
  Rep n;
};
```

$$value = n \times Radix^{Exponent}$$


…0000000000001010100.000000…

…0000000000**0101010**.000000…

```
int8_t{42}
```

…0000000000**0101010**.000000…

```
int8_t{42}
```

…0000000000**0101010**.000000…

```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```

…0000000000**0101010**.000000…

```
int8_t{42}
```

✗

```
scaled_integer<int8_t, power<1>>{42}
```

?

✓

```
scaled_integer<int8_t, power<1>>
a(float value)
{
    return value;
}
```

```
a(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz
  pop   {r4, pc}
```

```
scaled_integer<int8_t, power<1>>
a(float value)
{
    return value;
}
```

```
int8_t b(float value)
{
    return value * .5f;
}
```

```
a(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz
  pop   {r4, pc}
```

```
b(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz
  lsl   r0, r0, #24
  asr   r0, r0, #24
  pop   {r4, pc}
```

```
scaled_integer<int8_t, power<1>>
a(float value)
{
    return value;
}
```

```
int8_t b(float value)
{
    return value * .5f;
}
```

```
a(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz
  pop   {r4, pc}
```

```
b(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz
  lsl   r0, r0, #24
  asr   r0, r0, #24
  pop   {r4, pc}
```

godbolt.org/z/C3rVEh

```
scaled_integer<int   , power<1>>
a(float value)
{
    return value;
}
```

```
int    b(float value)
{
    return value * .5f;
}
```

```
a(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz
  pop   {r4, pc}
```

```
b(float):
  push  {r4, lr}
  mov   r1, #1056964608
  bl    __aeabi_fmul
  bl    __aeabi_f2iz


  pop   {r4, pc}
```

godbolt.org/z/FKugVN

…00000000**00010101**0.000000…

```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```

…00000000**00010101**0.000000…

```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```

…00000000**00101010**.000000…

```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```

```
scaled_integer<int, power<1>>{42}
```

…00000000**00101010**.000000…

```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```

```
scaled_integer<int, power<1>>{42}
```

?

```
namespace cnl::literals {
    template<char ... Chars>
    constexpr auto operator ""_c() noexcept;
}
```

```
namespace cnl::literals {
    template<char ... Chars>
    constexpr auto operator ""_c() noexcept;
}
```

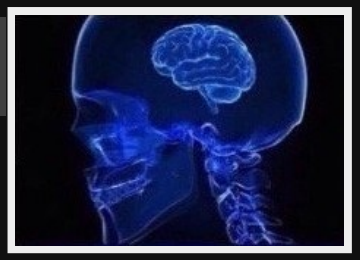
```
// constant<42>
auto forty_two = 42_c;

// scaled_integer<int, power<1>>{42}
auto a = scaled_integer{forty_two};
```

```
namespace cnl::literals {
    template<char ... Chars>
    constexpr auto operator ""_c() noexcept;
}
```

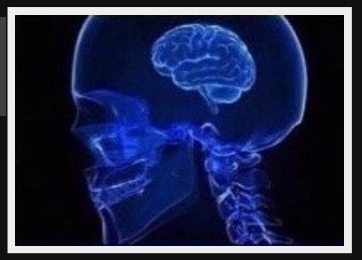
```
// constant<42>
auto forty_two = 42_c;

// scaled_integer<int, power<1>>{42}
auto a = scaled_integer{forty_two};
```

godbolt.org/z/x6PbFK

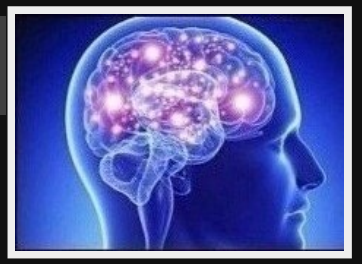
```
int8_t{42}
```



```
int8_t{42}
```

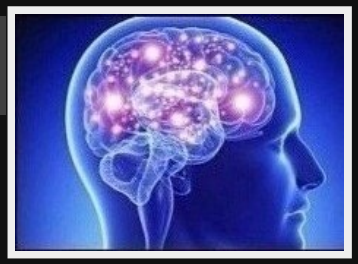


```
scaled_integer<int8_t, power<1>>{42}
```

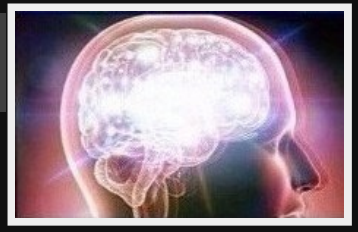


```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```

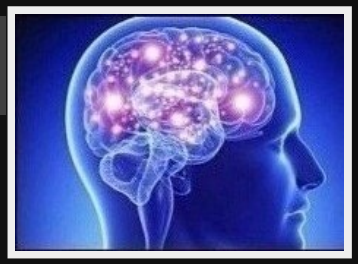


```
scaled_integer<int, power<1>>{42}
```

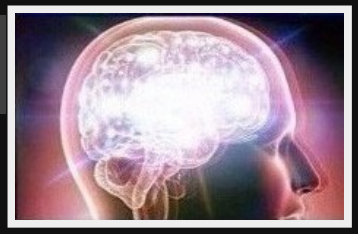


```
int8_t{42}
```

```
scaled_integer<int8_t, power<1>>{42}
```



```
scaled_integer<int, power<1>>{42}
```



```
scaled_integer{42_c}
```

Question: how do you *write* 'forty-two'?

Question: how do you *write* 'forty-two'?

Answer:

Question: how do you *write* 'forty-two'?
Answer: `scaled_integer{42_c}`

2.1. All Your Base

2.1. All Your Base

Question:

2.1. All Your Base

Question: how many bits in a megabyte?

1M

1M

1000

1M

1000×


1M
1000×1000

1M

1000×1000

1111101000

1M

1000×1000

1111101000×

1M

1000×1000

1111101000×1111101000

$$1111101000 \times 1111101000$$

$$1111101000 \times 1111101000$$
$$= 1111010000100100000 0$$

$$1111101000 \times 1111101000$$
$$= 1111010000100100000000$$

$$1111101000 \times 1111101000$$
$$= 1111010001001000000$$

```
// scaled_integer<int, power<3>>{1000}
auto k = scaled_integer{1000_c};

// scaled_integer<int, power<6>>{1000000}
auto m = k*k;
```

```
// scaled_integer<int, power<3>>{1000}
auto k = scaled_integer{1000_c};

// scaled_integer<int, power<6>>{1000000}
auto m = k*k;
```

godbolt.org/z/xWwhNz

Question: how many bits in a megabyte?

Question: how many bits in a megabyte?

Answer:

Question: how many bits in a megabyte?

Answer: 7

Question: how many bits in a megabyte?
Answer: 7+

Question: how many bits in a megabyte?
Answer: 7+7

Question: how many bits in a megabyte?
Answer: 7+7=14

Question: how many bits in a megabyte?

Answer: ~~7+7=14~~

1M

1M

1000

1M

1000×


1M
1000×1000

1M

1000×1000

=1000000

$$1000 \times 1000 = 1000000$$

$$1000 \times 1000 = 1000000$$

```
// scaled_integer<int, power<3, 10>>{1000}
auto k = scaled_integer<int, power<3, 10>>{1000};

// scaled_integer<int, power<6, 10>>{1000000}
auto m = k*k;
```

```
// scaled_integer<int, power<3, 10>>{1000}
auto k = scaled_integer<int, power<3, 10>>{1000};

// scaled_integer<int, power<6, 10>>{1000000}
auto m = k*k;
```

godbolt.org/z/zzg7BC

Question: how many bits in a megabyte?

Question: how many bits in a megabyte?
Answer:

Question: how many bits in a megabyte?
Answer: 1

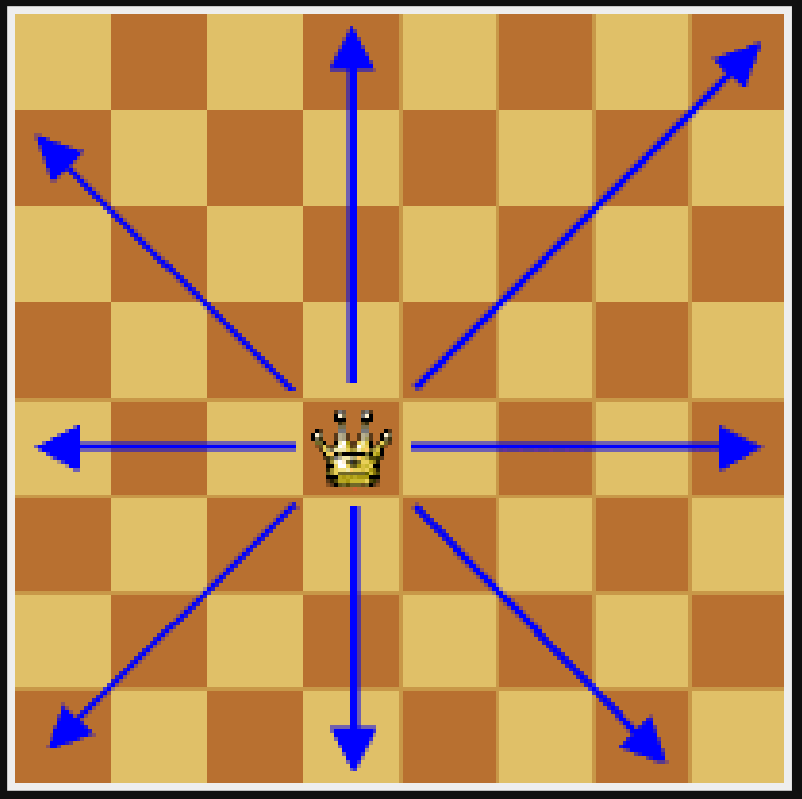
2.2. Root Planning

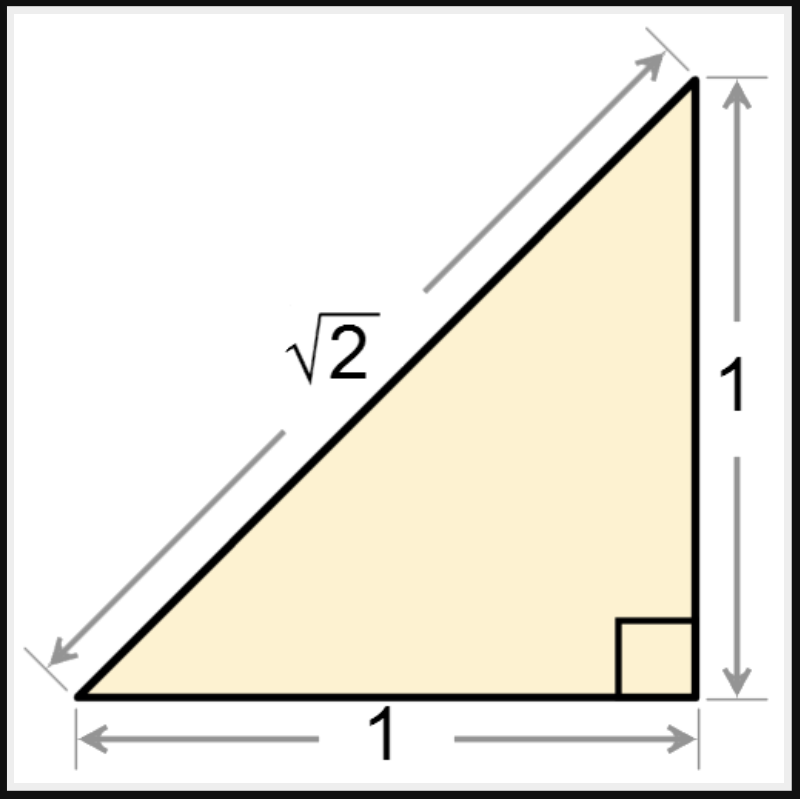
2.2. Root Planning

Question:

2.2. Root Planning

Question: how far does the queen really move in Chess?




√2
1
1

1.41421356237309504880168872420969807856967187 5…

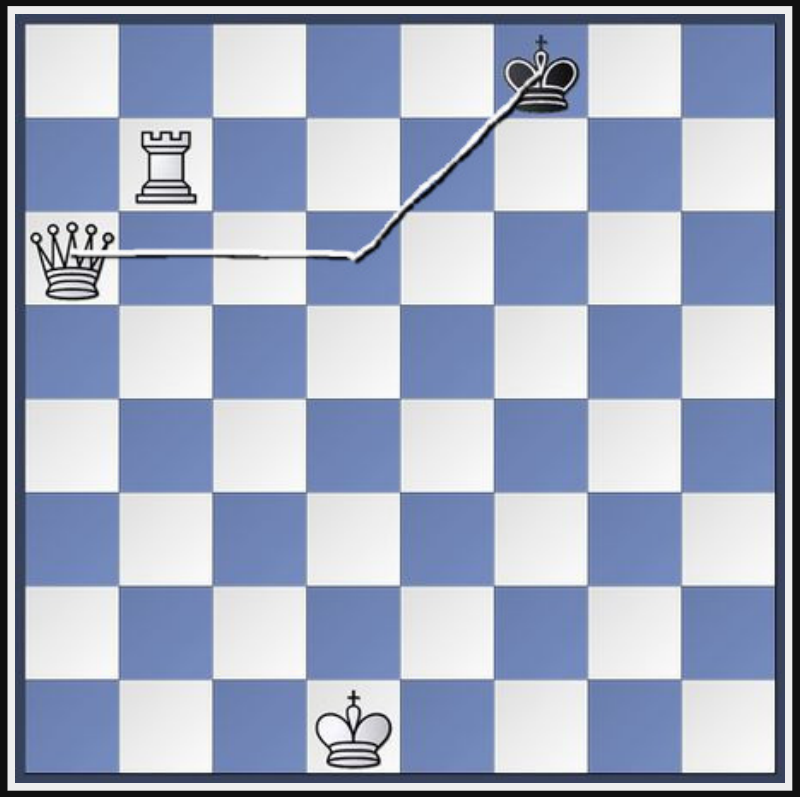
1.41421356237309504880168872420969807856967187 5…

…001.011010100000100111100110011001111111001 11011…

1.41421356237309504880168872420969807856967187…

…001.0110101000001001111001100110011111110011101…

sqrt2_v<scaled_integer<int, power<-30>>>

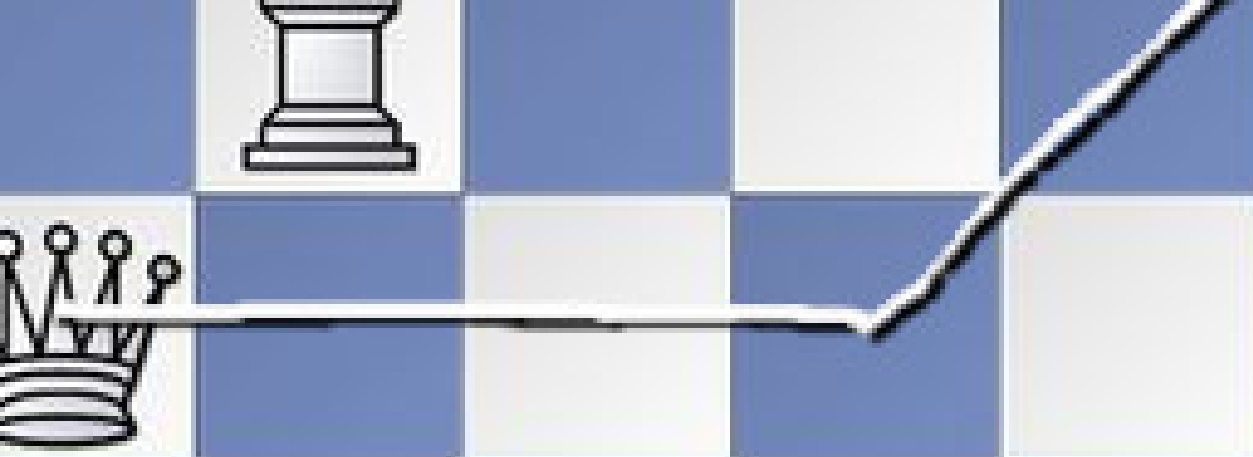


```
using Distance = scaled_integer<int, power<-24>>;

Distance queens_distance(int columns, int rows)
{
    columns = std::abs(columns);
    rows = std::abs(rows);
    int diagonal = std::min(columns, rows);
    int axial = std::max(columns, rows)-diagonal;
    return sqrt2_v<Distance>*diagonal+axial;
}
```

godbolt.org/z/9xcJWK

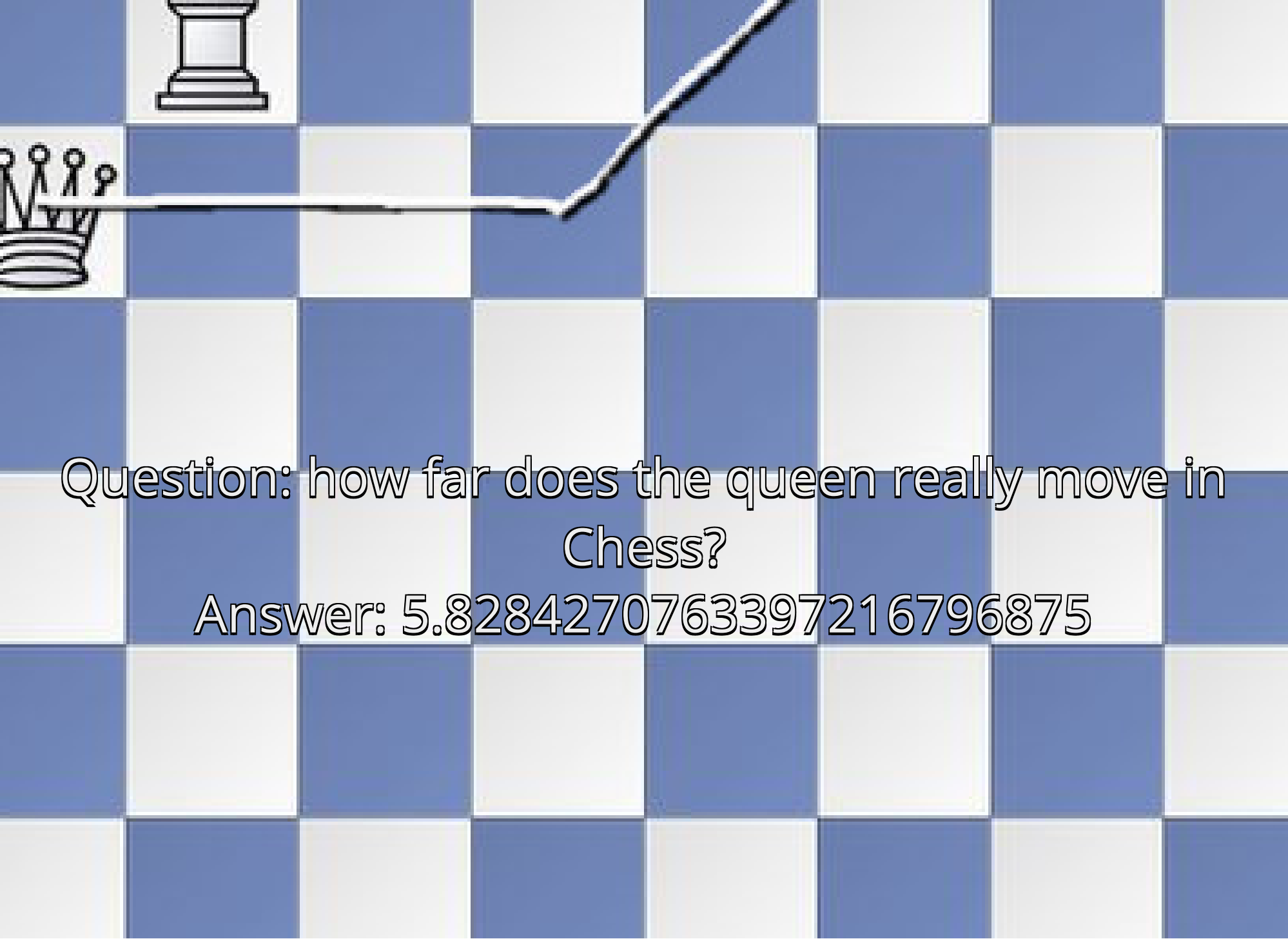
```
queens_distance(int, int):
        cmp     r0, #0
        rsblt   r0, r0, #0
        cmp     r1, #0
        rsblt   r1, r1, #0
        cmp     r0, r1
        movlt   ip, r0
        movge   ip, r1
        cmp     r0, r1
        rsbge   r0, ip, r0
        rsblt   r0, ip, r1
        ldr     r2, .L3
        mul     r3, r2, ip
        add     r0, r3, r0, lsl #24
        bx      lr
.L3:
        .word   23726566
```

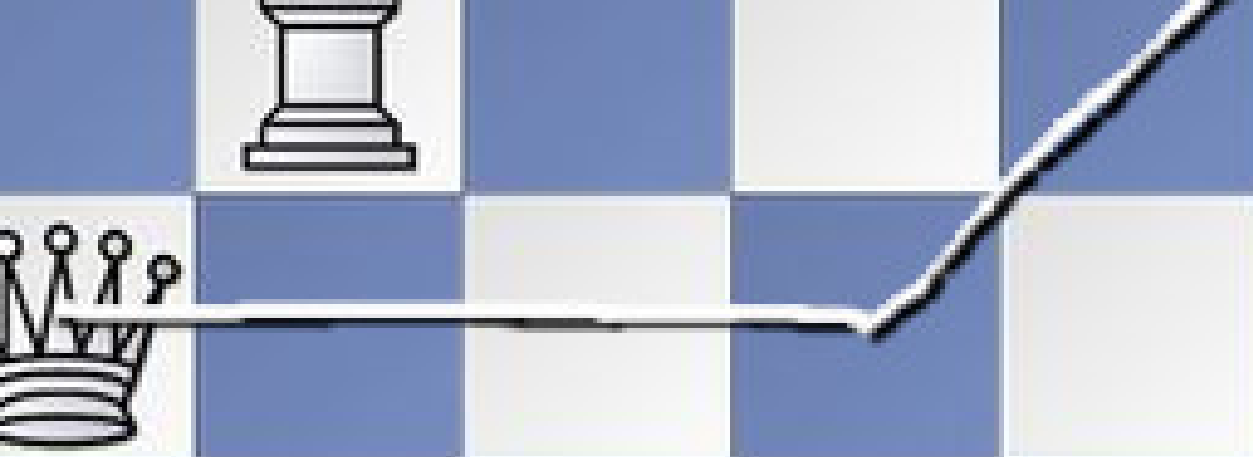

Question: how far does the queen really move in Chess?



Question: how far does the queen really move in Chess?

Answer:


Question: how far does the queen really move in Chess?
Answer: 5.828427076339721679687 5



Question: how far does the queen really move in Chess?

Answer: 5.828427076339721679687 5

godbolt.org/z/2f3KjT

3. Range-Based Four

(0/0)

o

3.0. Prescriptive

3.0. Prescriptive

Question:

3.0. Prescriptive

Question: what if I run out of bits?

…00001.01101010000010011110011001100111111110…

…00001.01101010000010011110011001100111111110…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100111111110…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100111111110…

…00001.01101010000010011110011001100111111110…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100000000…

…00001.01101010000010011110011001100111111110…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100000000…

```
auto root32 = root2 * 4;
```

…00001.01101010000010011110011001100111111110…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100000000…

```
auto root32 = root2 * 4;
```

…00101.10101000001001111001100110010000000…

…00001.01101010000010011110011001100111111110…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100000000…

```
auto root32 = root2 * 4;
```

…00101.10101000001001111001100110010000000…

godbolt.org/z/FcU6DR

…00001.01101010000010011110011001100111111110…

```
constexpr auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100000000…

```
constexpr auto root32 = root2 * 4;
```

```
<source>:8:33: error: overflow in constant expression [-fpermi
 constexpr auto root32 = root2 * 4;
                                 ^
```

godbolt.org/z/nhsEaK

```
c++ -std=c++17
    -Wall -Wextra -Werror
    -fsanitize=address,undefined
    all_your_source_files.cpp
```

```
c++ -std=c++17
    -Wall -Wextra -Werror
    -fsanitize=address,undefined
    all_your_source_files.cpp
```

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
auto root32 = root2 * 4;
```

```
c++ -std=c++17
    -Wall -Wextra -Werror
    -fsanitize=address,undefined
    all_your_source_files.cpp
```

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
auto root32 = root2 * 4;
```

```
cnl.h:2903:51: runtime error: signed integer overflow:
1518500249 * 4 cannot be represented in type 'int'
```

```
c++ -std=c++17
    -Wall -Wextra -Werror
    -fsanitize=address,undefined
    all_your_source_files.cpp
```

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
auto root32 = root2 * 4;
```

```
cnl.h:2903:51: runtime error: signed integer overflow:
1518500249 * 4 cannot be represented in type 'int'
```

wandbox.org/permlink/womtuT7zYstKiLhK

```
unsigned mu = 0;
cout << (mu-1);
```

```
unsigned mu = 0;
cout << (mu-1);
```

4294967295

```
unsigned mu = 0;
cout << (mu-1);
```

4294967295

```
int s = 1;
cout << (s << 31);
```

```
unsigned mu = 0;
cout << (mu-1);
```

4294967295

```
int s = 1;
cout << (s << 31);
```

-2147483648

```
unsigned mu = 0;
cout << (mu-1);
```

4294967295

```
int s = 1;
cout << (s << 31);
```

-2147483648

```
int ms = 100000;
cout << short(ms);
```

```cpp
unsigned mu = 0;
cout << (mu-1);
```

4294967295

```cpp
int s = 1;
cout << (s << 31);
```

-2147483648

```cpp
int ms = 100000;
cout << short(ms);
```

-31072

```
unsigned mu = 0;
cout << (mu-1);
```

```
4294967295
```

```
int s = 1;
cout << (s << 31);
```

```
-2147483648
```

```
int ms = 100000;
cout << short(ms);
```

```
-31072
```

wandbox.org/permlink/yKUMQNIRr9XZMVBU

```
// cnl/overflow_integer.h
namespace cnl {
   template<
      typename Rep,
      class OverflowTag=undefined_overflow_tag>
   class overflow_integer;
}
```

runtime error: execution reached an unreachable program point

runtime error: execution reached an unreachable program point

runtime error: execution reached an unreachable program point

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> s = 1;
cout << (s<<31);
```

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> s = 1;
cout << (s<<31);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> s = 1;
cout << (s<<31);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> ms = 100000;
cout << short(ms);
```

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> s = 1;
cout << (s<<31);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> ms = 100000;
cout << short(ms);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<unsigned> mu = 0;
cout << (mu-1);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> s = 1;
cout << (s<<31);
```

```
runtime error: execution reached an unreachable program point
```

```
overflow_integer<int> ms = 100000;
cout << short(ms);
```

```
runtime error: execution reached an unreachable program point
```

wandbox.org/permlink/7Mgbgw3AKcYeJzw0

Question: what if I run out of bits?

Question: what if I run out of bits?

Answer:

Question: what if I run out of bits?
Answer: don't.

3.1. Descriptive Range

3.1. Descriptive Range

Question:

3.1. Descriptive Range

Question: what if your numbers grow?

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$11111010000_4$$

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$1111101000^{4}$$

$$=1111010000100100000^{2}$$

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$1111101000^{4}$$

$$=1111010001001000000^{2}$$

$$=11101000110101001010010100010000000000000$$

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$1111101000^4$$

$$=11110100001001000000^2$$

$$=11101000110101001010010100010000000000000$$

```
// elastic_integer<10>{1000}
auto k = elastic_integer{1000_c};

// elastic_integer<20>{1000000}
auto m = k*k;

// elastic_integer<40>{1000000000000LL}
auto t = m*m;
```

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$1111101000^4$$

$$=11110100001001000000^2$$

$$=11101000110101001010010100010000000000000$$

```
// elastic_integer<10>{1000}
auto k = elastic_integer{1000_c};

// elastic_integer<20>{1000000}
auto m = k*k;

// elastic_integer<40>{1000000000000LL}
auto t = m*m;
```

godbolt.org/z/WKEkRk

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$1111101000^{4}$$

$$=11110100001001000000^{2}$$

$$=11101000110101001010010100010000000000000$$

```
// elastic_integer<10>{1000}
auto k = elastic_integer{1000_c};

// elastic_integer<20>{1000000}
auto m = k*k;

// elastic_integer<40>{1000000000000LL}
auto t = m*m;
```

godbolt.org/z/WKEkRk

```
// cnl/elastic_number.h
namespace cnl {
   template<int Digits, int Exponent>
   using elastic_number =
       scaled_integer<elastic_integer<Digits>, power<Exponent>>;

   namespace literals {
      template<char ... Chars>
      constexpr auto operator ""_elastic() noexcept;
   }
}
```

```
// cnl/elastic_number.h
namespace cnl {
   template<int Digits, int Exponent>
   using elastic_number =
       scaled_integer<elastic_integer<Digits>, power<Exponent>>;

   namespace literals {
      template<char ... Chars>
      constexpr auto operator ""_elastic() noexcept;
   }
}
```

```
// user code
using namespace cnl::literals;

// elastic_number<7, 3>
constexpr auto k = 1000_elastic;
```

```
// cnl/elastic_number.h
namespace cnl {
   template<int Digits, int Exponent>
   using elastic_number =
       scaled_integer<elastic_integer<Digits>, power<Exponent>>;

   namespace literals {
      template<char ... Chars>
      constexpr auto operator ""_elastic() noexcept;
   }
}
```

```
// user code
using namespace cnl::literals;

// elastic_number<7, 3>
constexpr auto k = 1000_elastic;
```

godbolt.org/z/tBYdGE

```
// cnl/elastic_integer.h
namespace cnl {
   template<int Digits, typename Narrowest=int>
   class elastic_integer;
}
```

$$1111101000^4$$

$$=11110100001001000000^2$$

$$=11101000110101001010010100010000000000000$$

```
// elastic_integer<10>{1000}
auto k = elastic_integer{1000_c};

// elastic_integer<20>{1000000}
auto m = k*k;

// elastic_integer<40>{1000000000000LL}
auto t = m*m;
```

godbolt.org/z/WKEkRk

```
// cnl/elastic_number.h
namespace cnl {
   template<int Digits, int Exponent>
   using elastic_number = ...;
}
```

$$1111101000^{4}$$

$$=1111010001001000000^{2}$$

$$=1110100011010100101001010001000000000000$$

```
// elastic_number<7, 3>{1000}
auto k = 1000_elastic;

// elastic_number<14, 6>{1000000}
auto m = k*k;

// elastic_number<28, 12>{1000000000000LL}
auto t = m*m;
```

```
// cnl/elastic_number.h
namespace cnl {
   template<int Digits, int Exponent>
   using elastic_number = ...;
}
```

$$1111101000^{4}$$

$$=11110100001001000000^{2}$$

$$=11101000110101001010010100010000000000000$$

```
// elastic_number<7, 3>{1000}
auto k = 1000_elastic;

// elastic_number<14, 6>{1000000}
auto m = k*k;

// elastic_number<28, 12>{1000000000000LL}
auto t = m*m;
```

godbolt.org/z/eRZs77

Question: what if your numbers grow?

Question: what if your numbers grow?

Answer:

Question: what if your numbers grow?
Answer: stretch!

3.2. Adaptive Range

3.2. Adaptive Range

Question:

3.2. Adaptive Range

Question: can't stretch / won't stretch?

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100100000000…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100100000000…

```
// scaled_integer<int, power<-28>>
auto root32 = root2 << 2_c;
```

…0000101.10101000001001111001100110010000000…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100100000000…

```
// scaled_integer<int, power<-28>>
auto root32 = root2 << 2_c;
```

…0000101.10101000001001111001100110010000000…

```
// elastic_number<31, -28>
auto root32 = root2 * 4_elastic;
```

…0000101.10101000001001111001100110010000000…

```
auto root2 = sqrt2_v<scaled_integer<int, power<-30>>>;
```

…00001.01101010000010011110011001100100000000…

```
// scaled_integer<int, power<-28>>
auto root32 = root2 << 2_c;
```

…0000101.10101000001001111001100110010000000…

```
// elastic_number<31, -28>
auto root32 = root2 * 4_elastic;
```

…0000101.10101000001001111001100110010000000…

godbolt.org/z/tJHPtR

```
auto f(scaled_integer<int, power<-30>> n) {
    return n << 2_c;
}
```

```
auto f(scaled_integer<int, power<-30>> n) {
    return n << 2_c;
}
```

```
f:
    bx     lr
```

```
auto f(scaled_integer<int, power<-30>> n) {
    return n << 2_c;
}
```

```
f:
    bx    lr
```

```
auto g(scaled_integer<int, power<-30>> n) {
    return n * 4_elastic;
}
```

```
auto f(scaled_integer<int, power<-30>> n) {
    return n << 2_c;
}
```

```
f:
    bx    lr
```

```
auto g(scaled_integer<int, power<-30>> n) {
    return n * 4_elastic;
}
```

```
g:
    bx    lr
```

```
auto f(scaled_integer<int, power<-30>> n) {
    return n << 2_c;
}
```

```
f:
    bx    lr
```

```
auto g(scaled_integer<int, power<-30>> n) {
    return n * 4_elastic;
}
```

```
g:
    bx    lr
```

godbolt.org/z/qSFyHN

Question: can't stretch / won't stretch?

Question: can't stretch / won't stretch?

Answer:

Question: can't stretch / won't stretch?
Answer: don't stretch.

4. Round Up

4.0. Conclusion

4.0. Conclusion

- 2∞+8

4.0. Conclusion

- 2∞+8
- ∞+1

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3
- `scaled_integer{42_c}`

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3
- `scaled_integer{42_c}`
- ~~14~~ 1

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3
- `scaled_integer{42_c}`
- ~~14~~ 1
- 5.8284270763397216796875

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3
- `scaled_integer{42_c}`
- ~~14~~ 1
- 5.8284270763397216796875
- don't.

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3
- `scaled_integer{42_c}`
- ~~14~~ 1
- 5.8284270763397216796875
- don't.
- stretch!

4.0. Conclusion

- 2∞+8
- ∞+1
- ∞+3
- 2∞+24
- 3
- `scaled_integer{42_c}`
- ~~14~~ 1
- 5.8284270763397216796875
- don't.
- stretch!
- don't stretch.

4.1. Thank You

Questions: ?

@JSAMcFarlane

fixed-point@john.mcfarlane.name

johnmcfarlane.github.io/slides/2019-accu