Before we .begin()
Before we .begin()

Does the code below compile? If so, how? If not, why?

(Wait… question is open only for C++ programmers with 0-3 years of experience)

```cpp
#include <utility>
#include <string>

int main() {
    std::string s1 = "hi", s2 = "bye";
    swap(s1, s2);

    int a = 3, b = 7;
    swap(a, b);
}
```
Complexity

What this talk is about (and what’s not)

Complexity

This talk is NOT about Algorithmic Complexity (no big ‘O’ in this talk!)
Complexity

This talk is NOT about Algorithmic Complexity (no big ‘O’ in this talk!)

But we do have a talk on Algorithmic Complexity - on Friday!

Complexity

This talk is about **C++ language complexity**, *with a broad definition for complexity:*

anything that *makes it hard for you to use C++*, or *to understand it*,
including things that *irritate or annoy* you, things that *waste your time*,
and language syntax that is *bug prone*, or *broken* in a way, or is *done easier*
in other languages.
About Me

Amir Kirsh

Lecturer
Academic College of Tel-Aviv-Yaffo and Tel-Aviv University

Developer Advocate

Co-Organizer of the CoreCpp conference and meetup group
Talk Origins

A graduated student of mine was interviewed for a C++ position and consulted me whether C++ is a right choice (as “there are other less complex languages”).

I convinced her.
Talk Origins

A graduated student of mine was interviewed for a C++ position and consulted me whether C++ is a right choice (as “there are other less complex languages”).

I convinced her.

She is now a C++ developer at Waves.com

We prepared this talk together for CoreCpp conference 2021 in Tel-Aviv.

And now you got me here :-)
Complexity

Isn’t it the name of the game? (of being a programmer...)

The Perils of Java Schools

Joel Spolsky, 2005

Lazy kids.
Whatever happened to hard work?
...
in the last decade a large number of otherwise perfectly good schools have gone 100% Java ...
The lucky kids of JavaSchools are never going to get weird segfaults trying to implement pointer-based hash tables. They’re never going to go stark, raving mad trying to pack things into bits.

Complexity

Is C++ complex?

Why?

Can it be less complex?

What makes C++ complex - for you?

Please check only the things that bother you in person, don’t check topics that you are not familiar with or don’t use at all:
What makes C++ complex - *for you*?

Please check only the things that bother you in person, don’t check topics that you are not familiar with or don’t use at all:

Let’s review the [questionnaire results](#)
Complexity

Why programming is complex?
What makes a software language complex?

- Too many ways for doing the same thing ("too many options")
What makes a software language complex?

- Too many ways for doing the same thing ("too many options")
- Too few ways for doing things ("hard to express yourself fluently")
- Lack of standard / documentation / proper examples / community
What makes a software language complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Too low level
  - Too high-level, abstract
  - Contradicting paradigms or rules
  - Rules are not intuitive or too complicated
- Complex problems
What makes a software language complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Too low level
  - Too high-level, abstract
  - Contradicting paradigms or rules
  - Rules are not intuitive or too complicated
- Complex problems
- New syntax, new stuff getting into the language
- Backward compatibility issues between language versions
What makes a software language complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Too low level
  - Too high-level, abstract
  - Contradicting paradigms or rules
  - Rules are not intuitive or too complicated
- Complex problems
- New syntax, new stuff getting into the language
- Backward compatibility issues between language versions
- Lack of proper tools
Complexity

Is C++ complex?

Why?

Can it be less complex?

What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Too low level
  - Too high-level, abstract
  - Contradicting paradigms or rules
  - Rules are not intuitive or too complicated
- Complex problems
- New syntax, new stuff getting into the language
- Backward compatibility issues between language versions
- Lack of proper tools
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)?
- Too few ways for doing things (“hard to express yourself fluently”)?
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Too low level
  - Too high-level, abstract
  - Contradicting paradigms or rules
  - Rules are not intuitive or too complicated
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Has quite a few low level features
  - Has quite a few high-level features
  - Has some contradicting paradigms or rules
  - Some rules are not intuitive or too complicated
- Complex problems?

Understanding and mastering C++'s complexity @ CppCon 2021
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Has quite a few low level features
  - Has quite a few high-level features
  - Has some contradicting paradigms or rules
  - Some rules are not intuitive or too complicated
- Deals frequently with complex problems
- New syntax, new stuff getting into the language
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Has quite a few low level features
  - Has quite a few high-level features
  - Has some contradicting paradigms or rules
  - Some rules are not intuitive or too complicated
- Deals frequently with complex problems
- New syntax, new stuff getting into the language
- Backward compatibility issues between language versions?
What makes C++ complex?

- Too many ways for doing the same thing (“too many options”)
- Too few ways for doing things (“hard to express yourself fluently”)
- Lack of standard / documentation / proper examples / community
- Complex model
  - Too Has quite a few low level features
  - Too Has quite a few high-level features
  - Has some contradicting paradigms or rules
  - Some rules are not intuitive or too complicated
- Deals frequently with complex problems
- New syntax, new stuff getting into the language
- Backward compatibility issues between language versions
- Lack of proper tools?

Understanding and mastering C++'s complexity @ CppCon 2021
What makes C++ complex?

- Complex model
  - Has quite a few low level features
  - Has quite a few high-level features
  - Has some contradicting paradigms or rules
  - Some rules are not intuitive or too complicated
- Deals frequently with complex problems
What makes C++ complex?

- Complex model
  - Has some contradicting paradigms or rules
  - Some rules are not intuitive or too complicated

and that’s what you hold against a language with >5M users and billions lines of code??
There is no silver bullet

Picture: https://www.infoq.com/articles/No-Silver-Bullet-Summary -- OOPSLA 2005, Montreal

“No Silver Bullet - Essence and Accident in Software Engineering” by Fred Brooks, 1986

Essence or Accident?
Let’s play...

```cpp
std::map<std::string, std::list<std::pair<Date, Price>>>::iterator quotesItr = stocks.find(id);
```
Essence or Accident?

std::map<std::string, std::list<std::pair<Date, Price>>>::iterator quotesItr
    = stocks.find(id);

auto quotesItr = stocks.find(id);
Essence or Accident?

What’s the problem here:

class Shape {
    Color color;
public:
    virtual void draw() const = 0;
    virtual void move(int diffX, int diffY) = 0;
};

Can the compiler *deduce* that a class *needs* a virtual destructor and provide one?
Can the compiler *deduce* that a class *needs* a virtual destructor and provide one?

What are the actual rules for “you must have a virtual destructor”?

class Shape {
    Color color;
public:
    virtual void draw() const = 0;
    virtual void move(int diffX, int diffY) = 0;
};

int main() {
    Rect r ({10, 10}, {20, 20});
    Shape* p = &r;
    p->draw();
}
template<
        class K,
        class V,
        size_t SIZE,
        class FetchFunc>
class Cache {
    struct Holder {
        V val;
        mutable typename list<K>::iterator posInList;
        Holder(V v, Date exp, typename list<K>::iterator pos)
          : val(v), posInList(pos), expiry(exp), accessed(Date()) {}  
        // ...
    private:
        Date expiry;
        mutable Date accessed;
    };  
    // ...
};

C++17:
http://coliru.stacked-crooked.com/a/e8eddd01f177a572
template<
class K, 
class V, 
size_t SIZE, 
class FetchFunc>
class Cache {
struct Holder {
    V val;
    mutable typename list<K>::iterator posInList;
    Holder(V v, Date exp, typename list<K>::iterator pos)
        : val(v), posInList(pos), expiry(exp), accessed(Date()) {} 
        // ...
    private:
        Date expiry;
        mutable Date accessed;
    };
    // ...
};

C++20:
http://coliru.stacked-crooked.com/a/47ce82fa46ffe3ba

See: Why don't I need to specify "typename" before a dependent type in C++20?
And: Why is _typename_ prefix still required in such a case in C++20?
Essence or Accident?

```cpp
int main() {
    int arr[] = {1, 2, 3, 3, 2, 1};
    std::set unique_values(std::begin(arr), std::end(arr));
    for(auto val : unique_values) {
        std::cout << val << ' ';  
    }
}
```

--

Someone = Andrei Zissu

Someone reported this on the @corecpp virtual meeting today:
```
std::array<int, 5> a{0, 1, 2, 3, 4};
auto s0 = std::set(a.begin(), a.end()); // 2 iterator elements
auto s1 = std::set(a.begin(), a.end()); // 5 int elements
```

Mistakes were made.
Essence or Accident?

```cpp
int main() {
    int arr[] = {1, 2, 3, 3, 2, 1};
    std::set unique_values(std::begin(arr), std::end(arr));
    for(auto val : unique_values) {
        std::cout << val << ' ';
    }
}
```

This is the same vector<int>{10, 20} vs vector<int>{10, 20} issue. Once you have that split, everything else follows.

For pointers `p` and `q`, I would expect vector(p, q) to give me a vector containing two pointers - since that's what that syntax looks like it asks for.

Essence or Accident?

How to store a value obtained from a vector `pop_back()`?
Essence or Accident?

How to store a value obtained from a vector `pop_back()`?

```cpp
auto val = vec.back();
vec.pop_back();

^ Maybe?
```

^ Maybe? Not really...
Essence or Accident?

How to store a value obtained from a vector `pop_back()`?

```cpp
auto val = std::move(vec.back());
vec.pop_back();
```
Essence or Accident?

```cpp
std::vector<bool> flags;
// ...

// need to toggle all flags
for(auto& flag: flags) {
    flag = !flag;
}
```
Essence or Accident?

```cpp
struct A {
    int foo(int) { return 7; }
};

struct B: A {
    int foo(float) { return 8; }
};

int main() {
    return B().foo(0); // 8 or 7 ?
}
```
 Essence or Accident?

```cpp
template<
    typename T>
std::enable_if_t<std::is_integral_v<T>> f(T t) {
    // integral version
}

template<
    typename T>
std::enable_if_t<std::is_floating_point_v<T>> f(T t) {
    // floating point version
}
```
Essence or Accident?

```cpp
template<typename T>
std::enable_if_t<std::is_integral_v<T>> f(T t) {
    // integral version
}

template<typename T>
std::enable_if_t<std::is_floating_point_v<T>> f(T t) {
    // floating point version
}
```

C++20

```cpp
void f(std::integral auto t) {
    // integral version
}

void f(std::floating_point auto t) {
    // floating point version
}
```
Essence or Accident?

```cpp
std::string s = "but I have heard it works even if you don’t believe in it";
s.replace(0, 4, "").replace(s.find("even"), 4, "only").replace(s.find(" don’t"), 6, "");
assert(s == "I have heard it works only if you believe in it");
```

Chaining is fixed, but only since C++17:


Very relevant to the pipe | syntax used by ranges
**C++ Principles (Stroustrup, C++ Design and Evolution)**

Static type system
- equal support for builtins and user defined types
- value and reference semantics

Resource and Memory management
- RAI Shiv scoped based
- No garbage collector

Efficient Object Oriented Programming
Flexible and efficient generic programming

Pay only for what you need
Direct access to OS and HW
Leave no room for a lower-level language below C++
* except assembler

See:
* The Design of C++, by Bjarne Stroustrup, 1994

---

**The Acronyms**
It's not complex… just go to the C++ acronym glossary by Arthur O'Dwyer
The Acronyms - partial list

Or join Bob Steagall’s talk here at CppCon 2021 on Friday afternoon

Watch also Kate Gregory’s great talk “It’s Complicated” from Meeting C++17
The Pyramid of C++ Knowledge

- Applicative C++ Developers
- Internal framework and utility maintainers
- Library and framework implementers
- Language Lawyers

The bare minimum to be a C++ programmer
The bare minimum to be a C++ programmer

the basic syntax, implicit casting rules, const correctness, constexpr, RAII, Rule of Zero, Rule of Three, operators overloading, static variables and static members, RValue and move semantics, Rule of Five, inheritance, polymorphism, multiple inheritance, virtual inheritance, exceptions, basic templates, variadic templates, forwarding reference and perfect forwarding, std containers, std algorithms, function objects, lambda, use of smart pointers
The bare minimum to be a C++ programmer

the basic syntax, implicit casting rules, const correctness, constexpr, RAII, Rule of Zero, Rule of Three, operators overloading, static variables and static members, RValue and move semantics, Rule of Five, inheritance, polymorphism, multiple inheritance, virtual inheritance, exceptions, basic templates, variadic templates, forwarding reference and perfect forwarding, std containers, std algorithms, function objects, lambda, use of smart pointers
reading code
browsing cppreference and stackoverflow

Being able to read is important, even crucial

Being able to read C++ code is even more important than writing

- know what you know
- know what you don’t know
- learn
Interviewing for a C++ junior position

Knows the bare minimum, or we are ready to train.
Interviewing for a C++ junior position

Knows the bare minimum, or we are ready to train.

Loves programming. Really, loves programming!

Smart and gets things done.
Interviewing for a C++ junior position

Knows the bare minimum, or we are ready to train.

Loves programming. Really, loves programming!

Smart and gets things done.

Implications of innocent ignorance

Well, I didn’t know that...
Implications of innocent ignorance

Well, I didn’t know that...

- Less elegant code (harder to maintain, harder to read)
- Less Generic code (could be written in a more generic way)
- Not being able to implement things
- Inefficient code
- Bug prone
- Actual bug!

Implications of innocent ignorance

Well, I didn’t know that...

- Less elegant code (harder to maintain, harder to read)
- Less Generic code (could be written in a more generic way)
- Not being able to implement things
- Inefficient code
- Bug prone
- Actual bug!
Improving your C++ level

Curiosity - in C++ everything has a reason, try to figure it out
**Improving your C++ level**

Curiosity - in C++ everything has a reason, try to figure it out

Read, listen, watch

^ Read Q&A in Stackoverflow
Improving your C++ level

Curiosity - in C++ everything has a reason, try to figure it out

Read, listen, watch

^ Read Q&A in Stackoverflow

^ Ask in Stackoverflow

^ Answer in Stackoverflow
Improving while answering in SO

Chrono timer in C++ to double

How can I create a timer that after a certain duration, it does something? So far I have been trying to use the % operation. I made a timer at the start of the function and subtracted the current time (now() function) then i % the difference by 5 because i want 5 seconds to pass:

```
(at the start of the program i defined start as high_resolution_clock::now())
dur = start-high_resolution_clock::now();
if(dur%5==0)
```

The error I've been getting is: no operator "==" matches these operands -- operand types are:

```
std::chrono::duration> == int
```

The solution is to use the `==` operator instead of `%` for comparison.
Chrono timer in C++ to double

You are looking for something like:

```cpp
#include <chrono>

int main()
    
    using std::chrono::high_resolution_clock;
    using std::chrono::duration<double>;
    using std::chrono::seconds;

    int main()
    
    int main()
    
    auto time_passed = start - high_resolution_clock::now();
    auto time_passed = start - high_resolution_clock::now();
    if (time_passed / duration<double>()) 
    if (time_passed / duration<double>()) {
        // do your thing
        // do your thing
    }
    // ...
    // ...
```
Chrono timer in C++ to double

1. How can I create a timer that after a certain duration, it does something? So far I have been trying to use the % operation. I made a timer at the start of the function and subtracted the current time (now() function) then I % the difference by 5 because I want 5 seconds to pass:

```cpp
(t() == 0) {
    start = high_resolution_clock::now();
    duration<double> dur = start - high_resolution_clock::now();
    if(dur%==0)
    }
```

The error I’ve been getting is: no operator “==” matches these operands -- operand types are:
std::chrono::duration> == int

What is the first error you get? I get the error about operator==, but it isn’t the first error.

Howard Hinnant Mar 1 '20 at 17:16

@AsteroidsWithWings would try. Having now Howard Hinnant on the page makes me a bit cold feet :-) at least if I’m wrong I’ll have someone to watch over.

Amir Kirsh Mar 1 '20 at 17:35
Chrono timer in C++ to double

How can I create a timer that after
how will I use the % operation. I made a t
(now)(function) then i % the differ-
(at the start of the program i
duration<double> dur = start-h
if((dur%==0))

the error I've been getting is: no oj
std::chrono::duration> == int

c++ chrono

What is the first error you get? I get the error about operator==, but it isn’t the first error. –
Howard Hinnant Mar 1 ’20 at 17:54

Chrono timer in C++ to double

How can I create a timer that after
to use the % operation. I made a t
(now)(function) then i % the differ-
(at the start of the program i
duration<double> dur = start-h
if((dur%==0))

the error I've been getting is: no oj
std::chrono::duration> == int

c++ chrono

You're doing fine. :-) But can you rewrite it without using .count() ? – Howard Hinnant Mar 1 ’20 at 17:54
Chrono timer in C++ to double

How can I create a timer that after a certain duration to use the \% operation. I made a timer at the start (now) function then I \% the difference by 5 because the error I've been getting is: no operator "\%" matches these operands -- operand types are:

```
std::chrono::duration> == int
```

What is the first error you get? I get the error about operator==, but it isn't the first error.

Howard Hinnant Mar 1 '20 at 18:01

You are looking for something like:

```
#include <chrono>

int main()
{
    using namespace std::chrono_literals;
    using std::chrono::high_resolution_clock;
    auto start = high_resolution_clock::now();
    bool condition = true;
    while (condition) {
        auto time_passed = high_resolution_clock::now() - start;
        if (time_passed % 5s == 0s) {
            // do your thing every 5 seconds
        }
    }
}
```

Explaination

The modulo

The modulo operation for duration called with: duration % x expects x to be either of the two:

- another duration, in which case chrono does the job for you in getting the common type of duration and x, that would allow modulo.

Much better! That was my upvote btw. Now you just need to clean up your explanation a little. :-) – Howard Hinnant Mar 1 '20 at 18:01
Improving your C++ level

Remember, it’s a never ending mission

Thank you!

```cpp
void conclude(auto greetings) {
    while(still_time() && have_questions()) {
        ask();
    }
    greetings();
}

conclude([]{ std::cout << "Thank you!"; });
```
Other Essence or Accident

out due to lack of time

Essence or Accident?

```cpp
template<class T, long Numerator, long Denominator, long MultNum, long MultDenom>
auto constexpr operator*(Aggregator<T, Numerator, Denominator> a, std::ratio<MultNum, MultDenom> n) {
    if constexpr(Numerator*MultNum != Denominator*MultDenom) {
        return Aggregator<T, Numerator * MultNum, Denominator * MultDenom> { a };
    } else {
        return a.unsafe_multiply(n);
    }
}

Source: The Point Challenge  https://www.youtube.com/watch?v=wNGEt1BSCLY
```
Essence or Accident?

Implement methods for rotating the x,y,z fields in Pixel struct below:

```cpp
struct Pixel {
    int x;
    int y;
    int z;
};
```

```cpp
class Permutation {
    std::array<int Pixel::* , 3> permutation;
    constexpr Permutation(int Pixel::* a, int Pixel::* b, int Pixel::* c)
        : permutation{a, b, c} {}  
public:
    static constexpr Permutation xzy() { return {&Pixel::x, &Pixel::z, &Pixel::y}; }  
    constexpr Pixel permutate(Pixel p) const {
        Pixel permutated;
        permutated.x = p.*permutation[0];
        permutated.y = p.*permutation[1];
        permutated.z = p.*permutation[2];
        return permutated;
    }
};
```

http://coliru.stacked-crooked.com/a/298a6e5a89e10a28