

accu
conference
2024

Symmetry in Code

Should We Care?

Victor Ciura

Symmetry in Code Should We Care?

ACCU

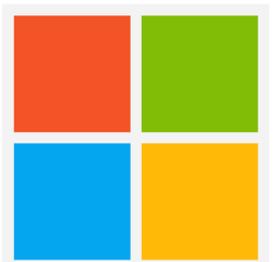
April 2024

 @ciura_victor

 @ciura_victor@hachyderm.io

 @ciuravictor.bsky.social

Victor Ciura
Principal Engineer
M365 Substrate



Why should we be concerned with symmetry? Symmetry is fascinating to the human mind and everyone likes objects or patterns that are in some way symmetrical. It is an interesting fact that nature often exhibits certain kinds of symmetry in the objects and phenomena in our Universe.

We have, in our minds, a tendency to accept symmetry as some kind of perfection. Yet it so often eludes us...

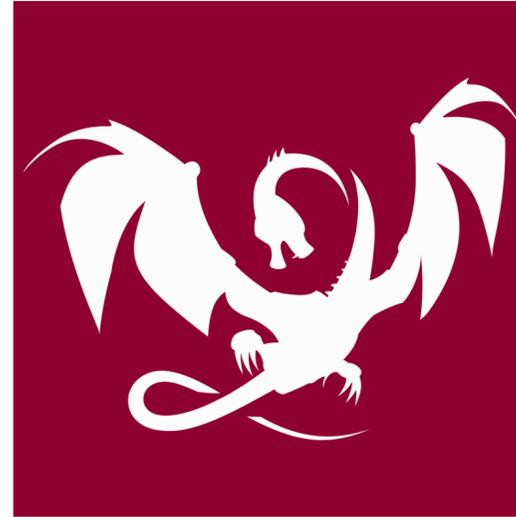
Let's look at code and see what interesting properties emerge from various kinds of symmetries. A quest for the 'Character of Code', following Richard Feynman's awe-inspiring take on physical laws.

We'll be looking to identify patterns in code, interested to see when such patterns exhibit some sort of symmetry that is advantageous in some way for reliability, performance, maintenance and discoverability.

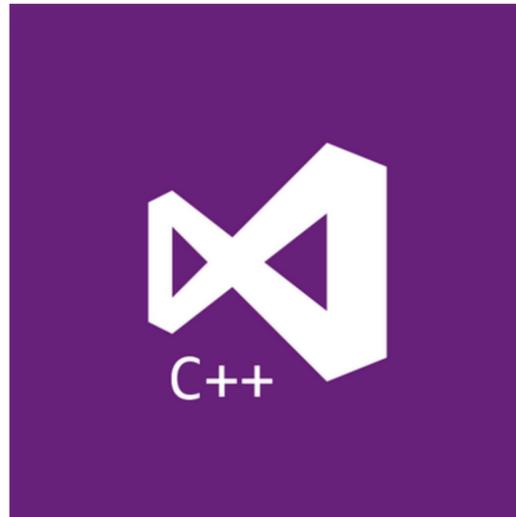
About me



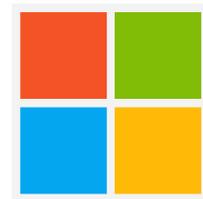
Advanced Installer



Clang Power Tools



Visual C++



M365 Substrate



@ciura_victor

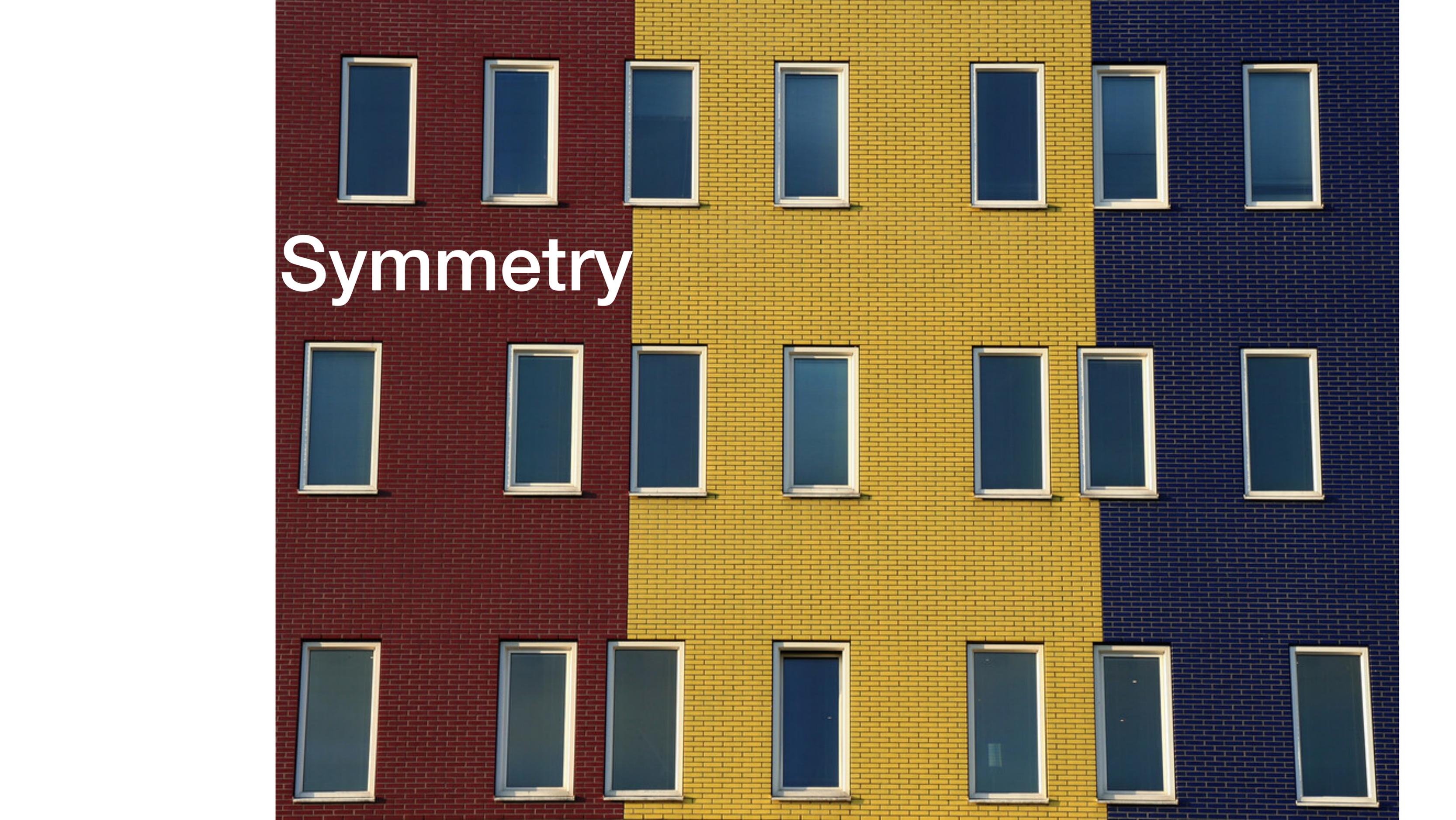


@ciura_victor@hachyderm.io

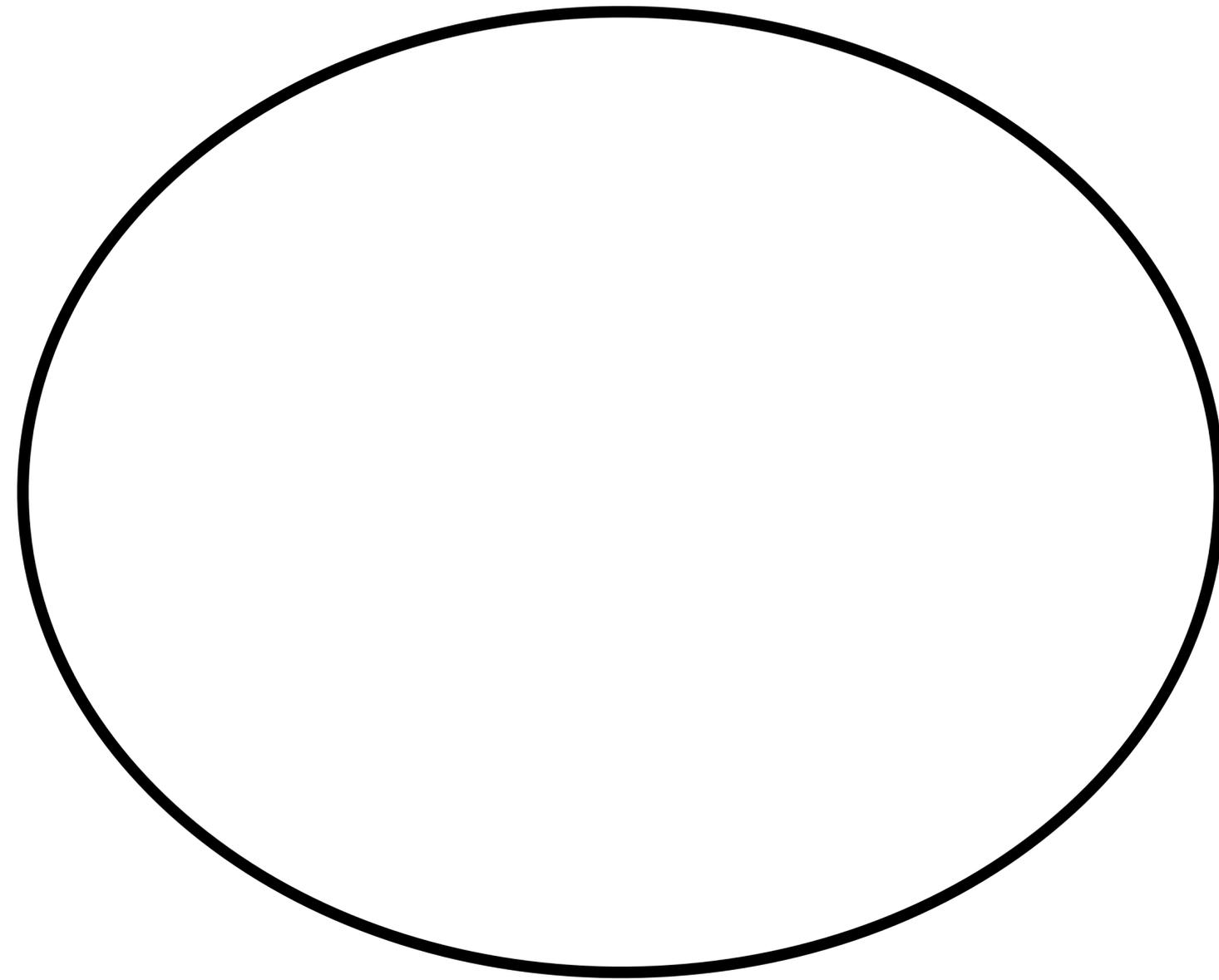


@ciuravictor.bsky.social



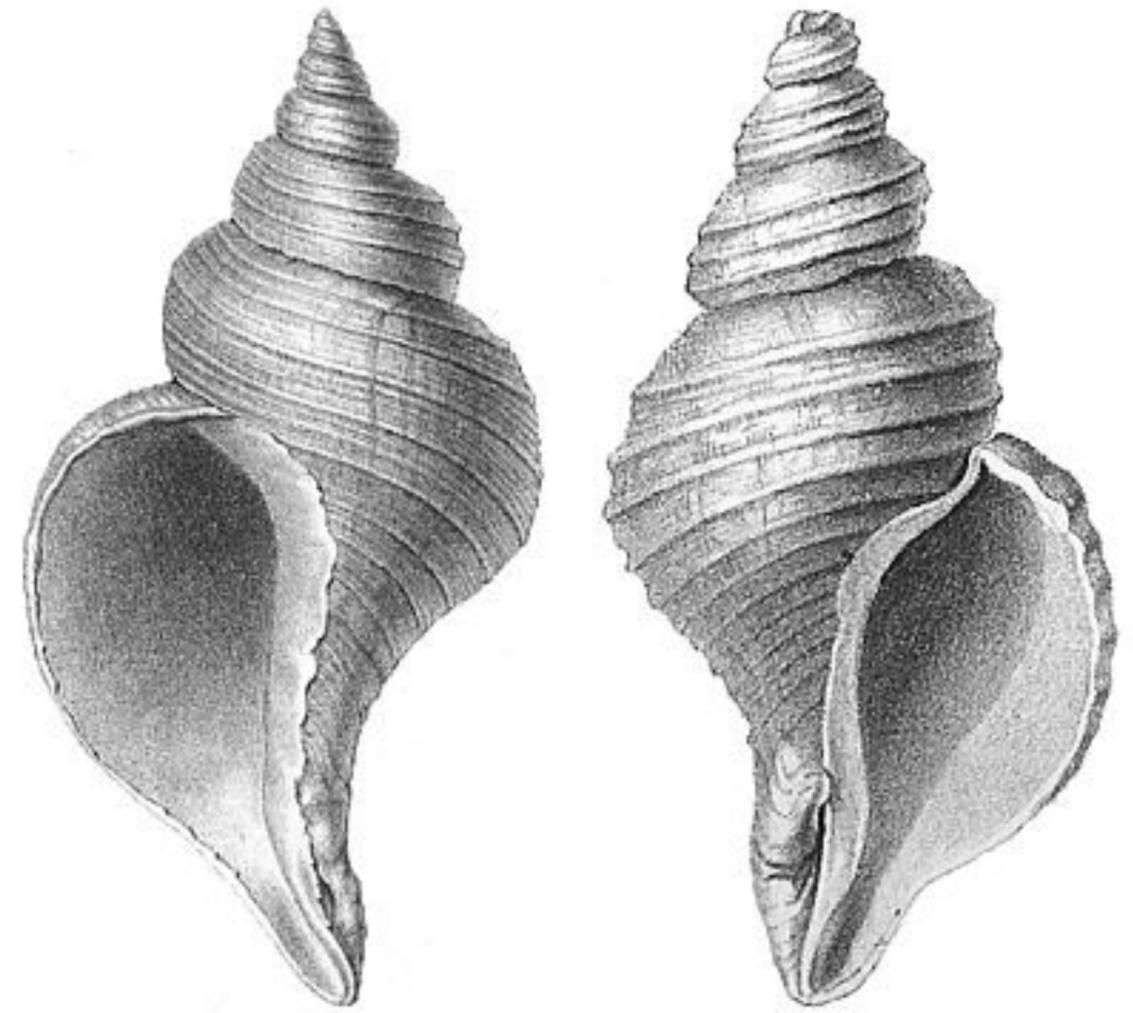
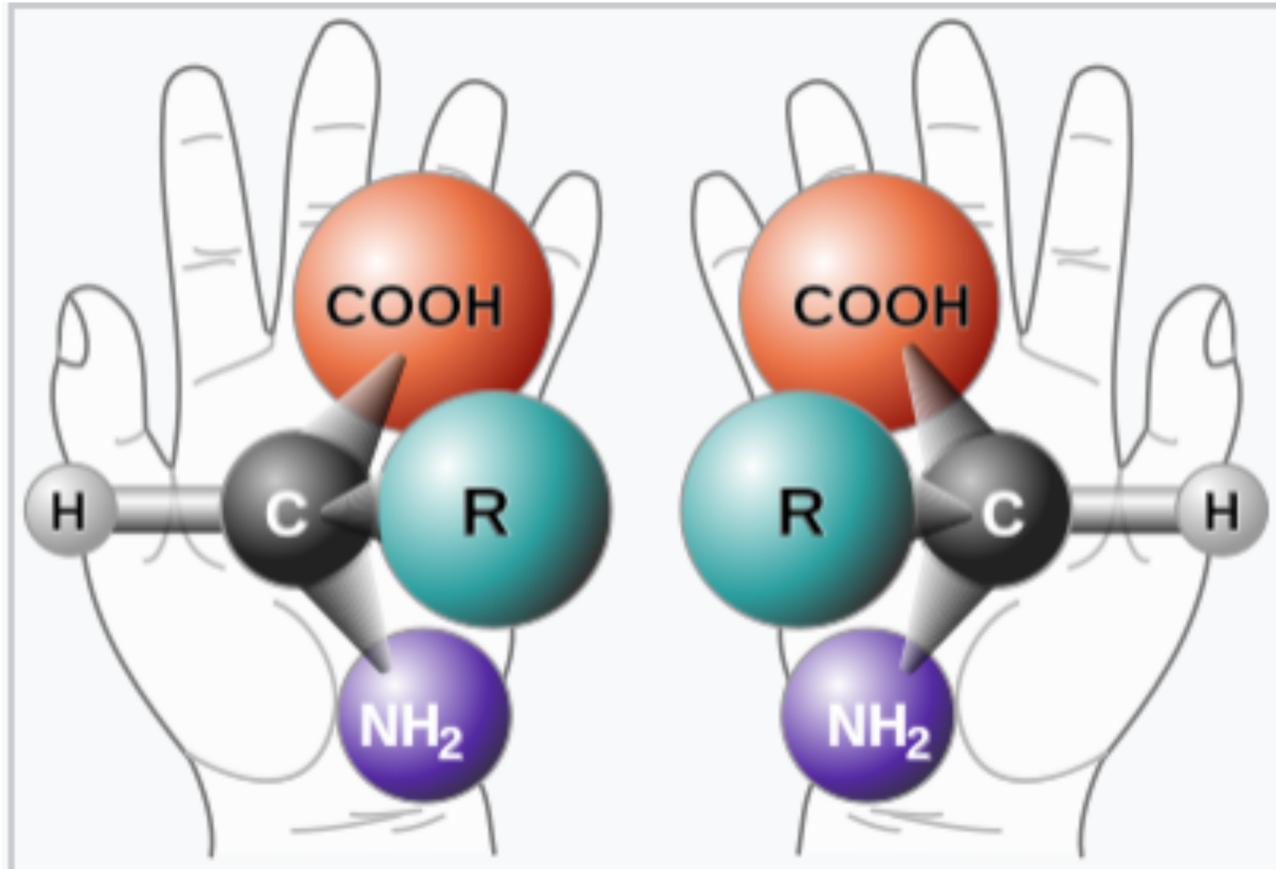
A photograph of a brick wall divided into three vertical sections of different colors: red on the left, yellow in the middle, and blue on the right. Each section contains three rectangular windows with white frames, arranged in a vertical column. The windows are evenly spaced and aligned horizontally across the three sections. The word "Symmetry" is written in white, sans-serif font across the middle of the red section.

Symmetry

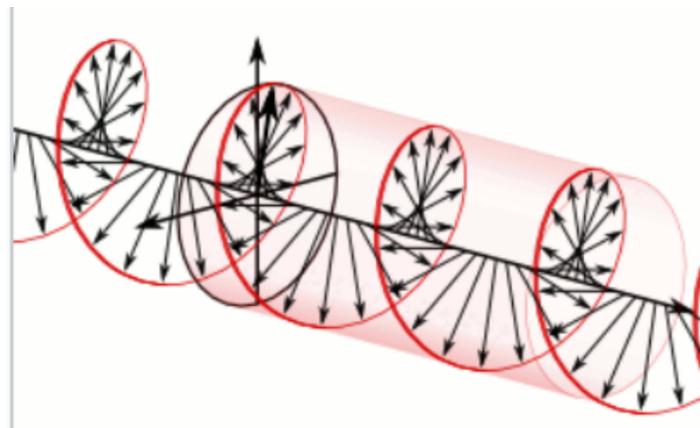


In fact, it is like the old idea of ancient $\zeta R \in \in K S$ that circles were perfect, and it was rather horrible for them to believe that the planetary orbits were not circles, but only nearly circles.

Chirality

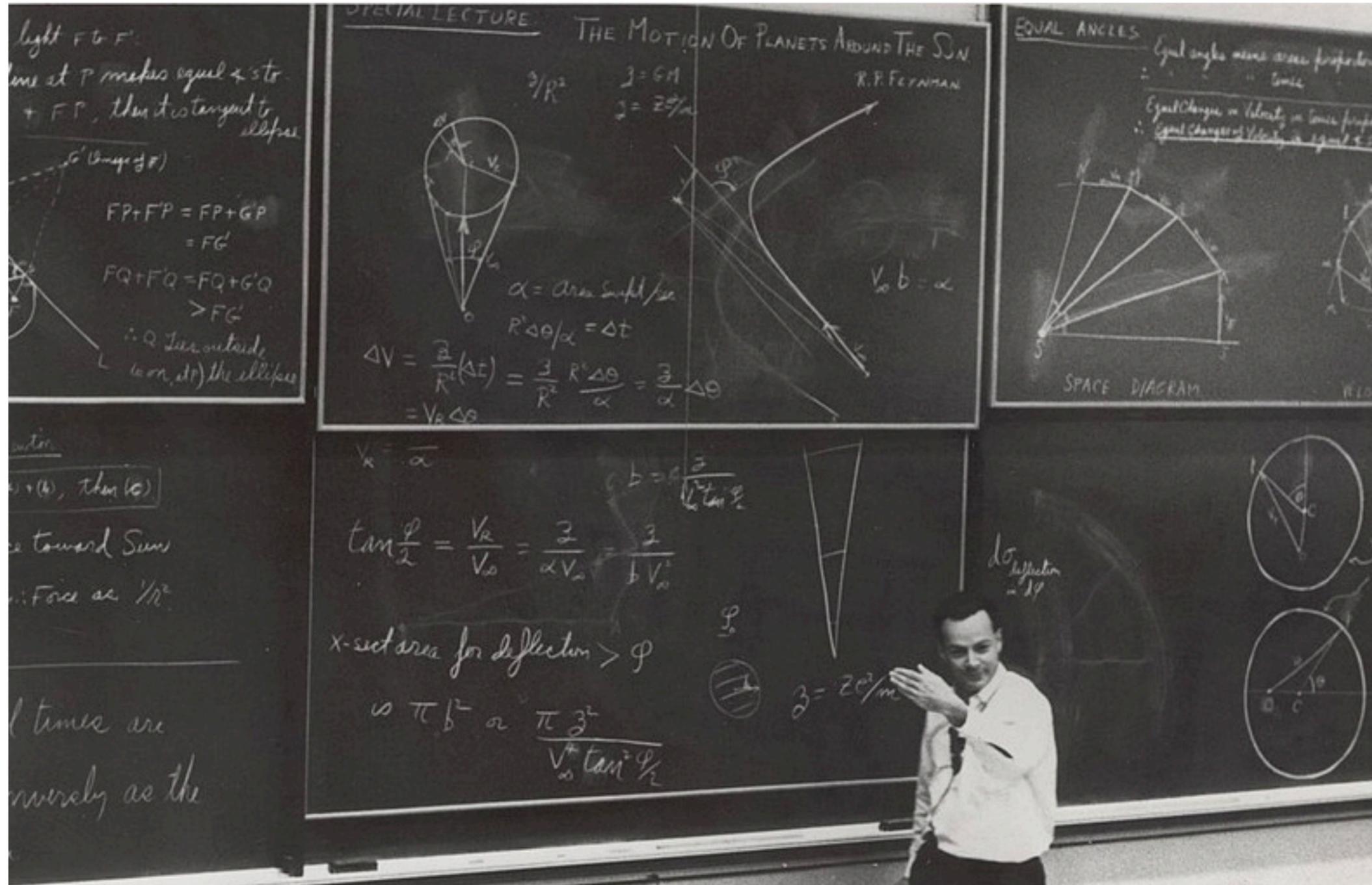


(left-handed) *Neptunea angulata* | (right-handed) *Neptunea despecta*



rotation of plane polarized light by chiral substances

Richard Feynman - The Character of Physical Law (1964)



Symmetry in Physical Laws

Translation in Space
Translation in Time
Rotation in Space
Uniform Vel in Straight line (Lorentz Trans.)
Reversal of Time
Reflection of Space
Replacement of one atom by another
Quant. Mech. Phase
Matter - Antimatter

Richard Feynman - The Character of Physical Law
Part 4: Symmetry in Physical Laws

[youtube.com/watch?v=tGsYbK-Chkg](https://www.youtube.com/watch?v=tGsYbK-Chkg)

Symmetry beyond geometry

Symmetry goes way beyond simple geometrical shapes & patterns.

Symmetry is not just about **observing** the properties of objects, but also for *transformations*:

- what can you **do** to a symmetrical object so it can "look" the same

He's the first mathematician to study symmetry for non-geometric entities (eg. equations, functions, polynomials, groups).





"Are elegant equations more likely to be right than inelegant ones?"

"Beauty is a very successful criterion for choosing the right theory"

Beauty, truth and ... physics?

1,527,719 views | Murray Gell-Mann | TED2007 • March 2007

ted.com/talks/murray_gell_mann

$$\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_z}{\partial z} = 4\pi\rho \quad (1)$$

$$\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z} = 0 \quad (2)$$

$$\left. \begin{aligned} \frac{\partial E_x}{\partial x} - \frac{\partial E_y}{\partial y} + \frac{1}{c} \dot{B}_z &= 0 \\ \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} + \frac{1}{c} \dot{B}_x &= 0 \\ \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} + \frac{1}{c} \dot{B}_y &= 0 \end{aligned} \right\} \quad (3)$$

$$\left. \begin{aligned} \frac{\partial B_x}{\partial y} - \frac{\partial B_y}{\partial x} - \frac{1}{c} \dot{E}_z &= \frac{4\pi}{c} j_z \\ \frac{\partial B_y}{\partial z} - \frac{\partial B_z}{\partial y} - \frac{1}{c} \dot{E}_x &= \frac{4\pi}{c} j_x \\ \frac{\partial B_z}{\partial x} - \frac{\partial B_x}{\partial z} - \frac{1}{c} \dot{E}_y &= \frac{4\pi}{c} j_y \end{aligned} \right\} \quad (4)$$

Original form

$$\nabla \cdot \mathbf{E} = 4\pi\rho \quad (1)$$

$$\nabla \cdot \mathbf{B} = 0 \quad (2)$$

$$\nabla \times \mathbf{E} + \frac{1}{c} \dot{\mathbf{B}} = 0 \quad (3)$$

$$\nabla \times \mathbf{B} - \frac{1}{c} \dot{\mathbf{E}} = \frac{4\pi}{c} \mathbf{j} \quad (4)$$

Simplified using rotational symmetry

$$\partial_\nu F^{\mu\nu} = \frac{4\pi}{c} j^\mu \quad (1 \text{ and } 4)$$

$$\epsilon^{\mu\nu\kappa\lambda} \partial_\nu F_{\kappa\lambda} = 0 \quad (2 \text{ and } 3)$$

Further simplified using the symmetry of special relativity

The Shape of A Program

The screenshot shows a video player interface. The main content area displays a presentation slide with a diagram and text. The diagram consists of a vertical purple bar with four downward-pointing arrows. In the middle of this bar are two overlapping circles: a light blue circle on the left labeled 'e' and a light purple circle on the right labeled 'f'. The text to the right of the diagram reads: "If we expand e and f to share entrances and exits, they remain alternative possibilities." A large play button is centered over the text. To the right of the slide is a video inset showing a woman with glasses, Lisa Lippincott, speaking. Above her is the text "C++ now" and "2018 MAY 7 - 11 cppnow.org". Below her is the text "Lisa Lippincott" and "The Shape of a Program". At the bottom of the video player, there is a control bar with a play button, a volume icon, a progress bar showing "0:00 / 1:15:07", a pause button, a CC icon, a settings gear, a full screen icon, and a logo for "JET BRAINS".

C++ now

2018
MAY 7 - 11
cppnow.org

na exits, they
possibilities.

e f

If we expand e and f to share entrances and exits, they remain alternative possibilities.

Lisa Lippincott

The Shape of a Program

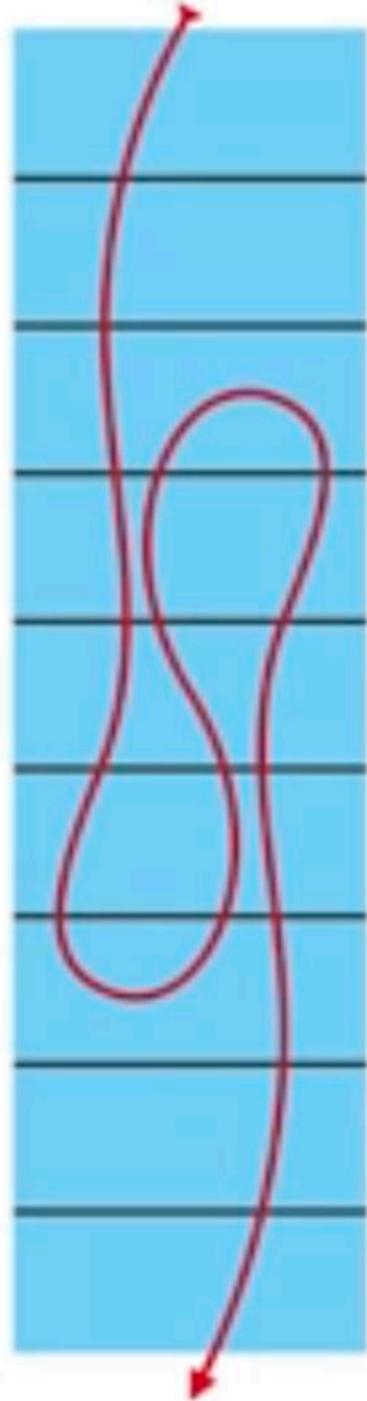
Video Sponsorship Provided By: JET BRAINS

0:00 / 1:15:07

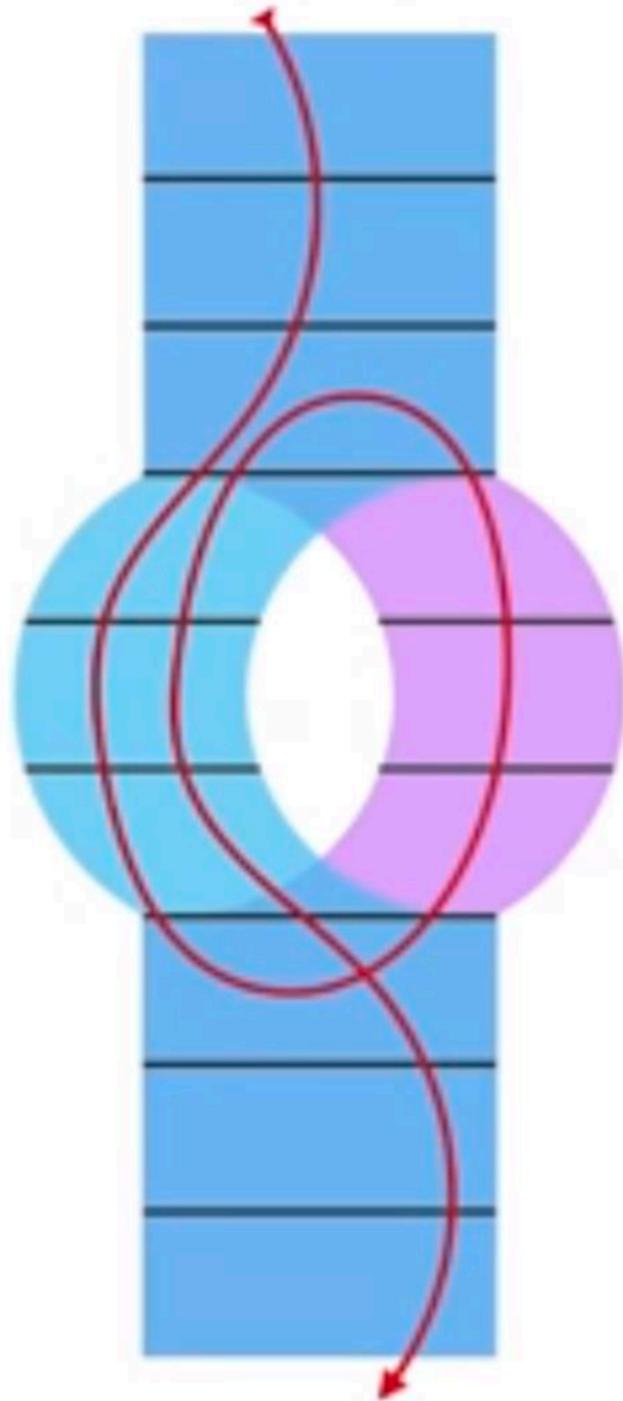
youtube.com/watch?v=QFIOE1jKv30

The Shape of A Program

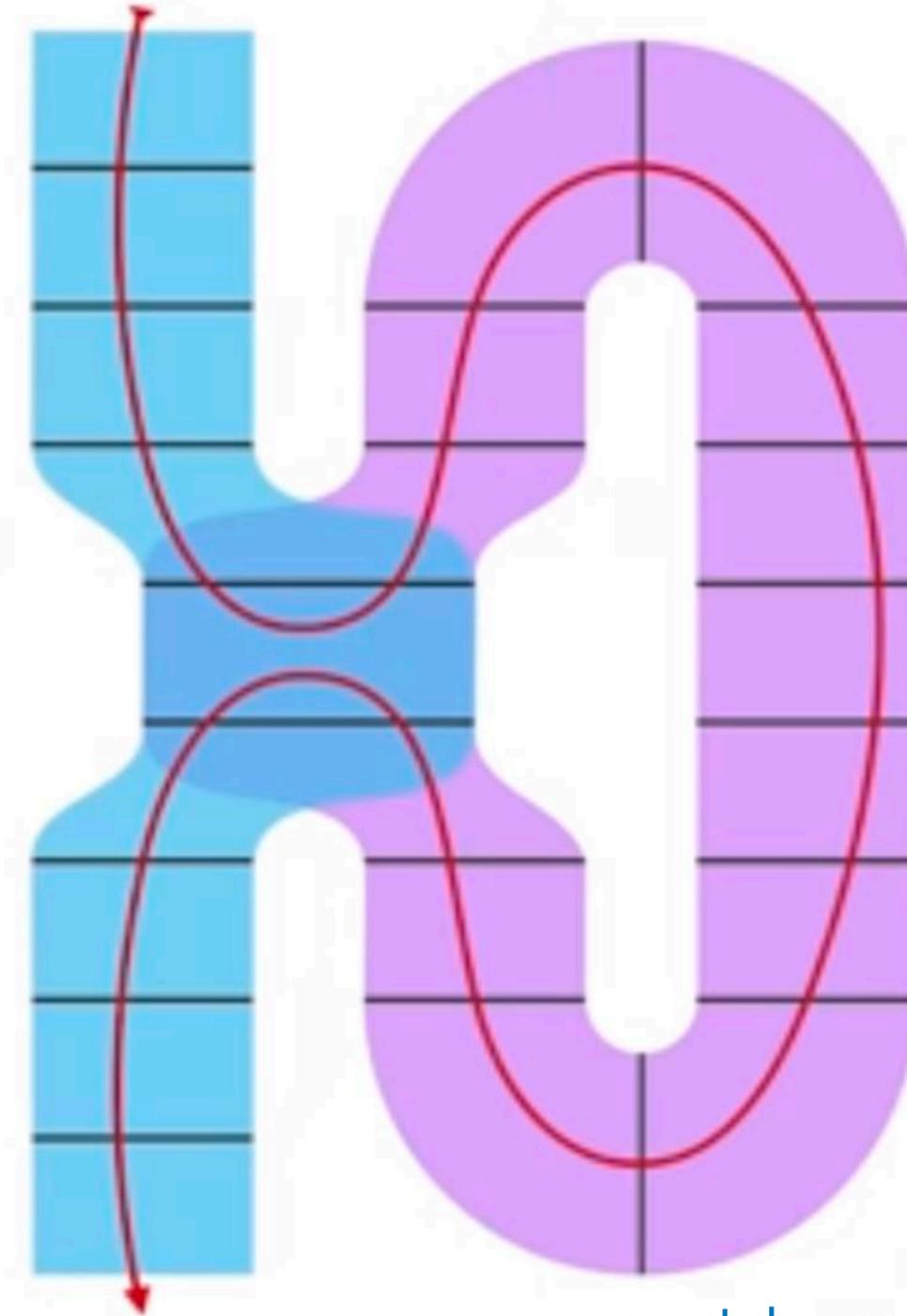
Sequence



Branch



Loop



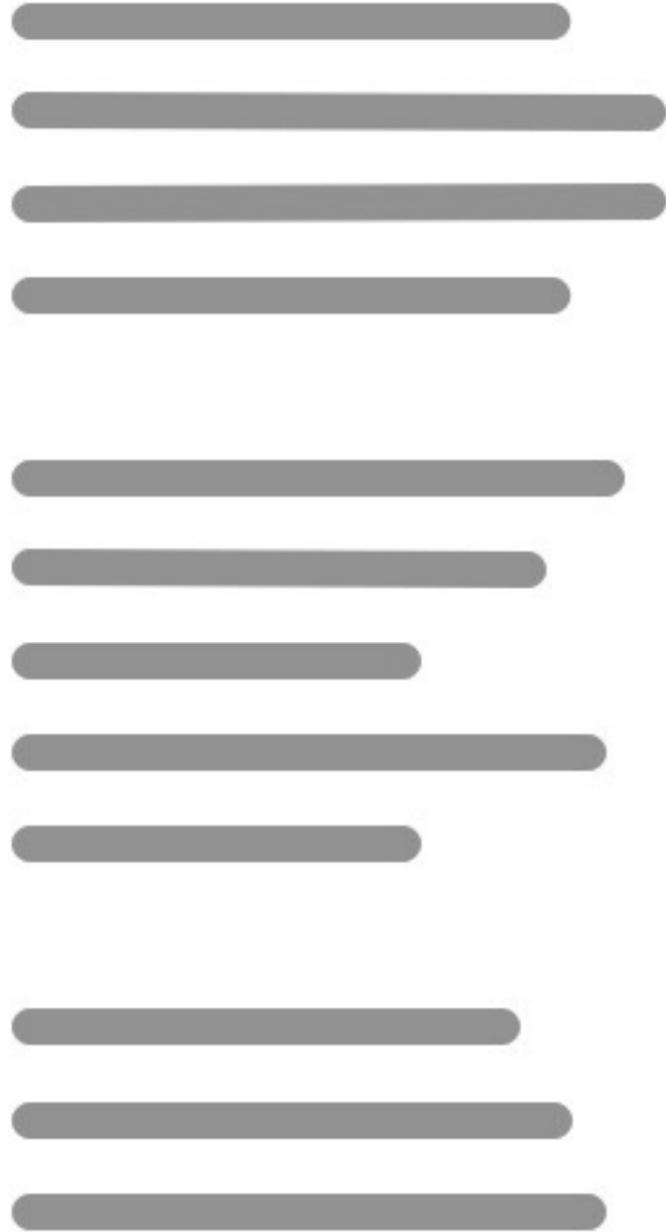
youtube.com/watch?v=QFIOE1jKv30

Shapes of Code



The saw

fluentcpp.com/2020/01/14/the-shapes-of-code/



The paragraphs

```
// ████████████████████  
████████████████████  
████████████████████  
████████████████████  
████████████████████
```

```
// ████████████████████████████  
████████████████████████  
████████████████████  
██████████████████  
████████████████████  
██████████████████
```

```
// ██████████  
██████████████████  
██████████████████  
██████████████████
```

The paragraphs with headers

Shapes of Code

```
if   
    
    
    
    
    
else  
  
```

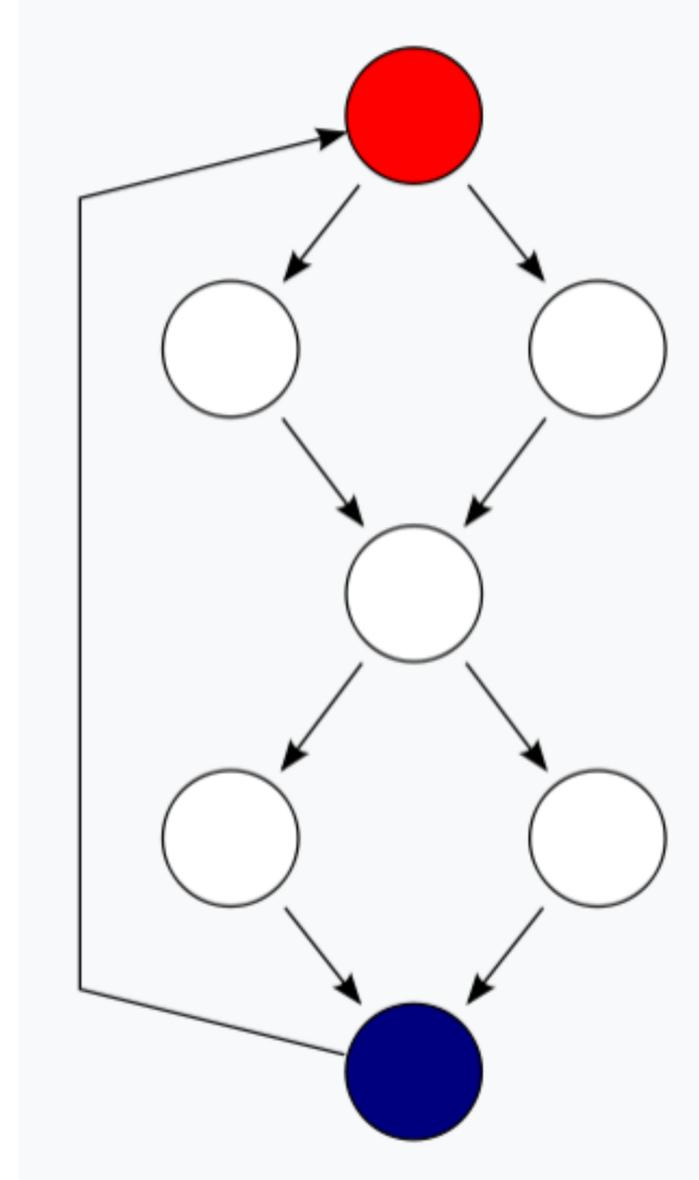
```
if   
    
else  
    
    
    
    
  
```

The unbalanced `if` blocks

Cyclomatic Complexity

```
int func()
{
  if (c1())
    f1();
  else
    f2();

  if (c2())
    f3();
  else
    f4();
}
```



wikipedia.org/wiki/Cyclomatic_complexity

The Shape of A Program

The image shows a YouTube video player interface. The main video area displays the title "THE SHAPE OF A PROGRAM" in large white letters on a black background. In the top right corner of the video frame, the text "cppcon | 2018" is visible, with "THE C++ CONFERENCE • BELLEVUE, WASHINGTON" underneath. To the right of the main video area, there is a smaller video thumbnail showing a man, James McNellis, speaking at a podium. Below the thumbnail, the name "JAMES McNELLIS" is written in white. Underneath the name, the title "The Shape of a Program" is displayed. At the bottom of the video player, there is a control bar with various icons: a play button, a volume icon, a progress bar showing "0:07 / 5:06", the name "Lisa Lippincott", a pause button, a closed captions icon, a settings gear icon, a share icon, a download icon, and a full screen icon.

youtube.com/watch?v=P2IxGnbDkDI

Program Complexity ?

```
int main()
{
// Seed with a real random value, if available
std::random_device r;

// Choose a random mean between 1 and 6
std::default_random_engine e1(r());
std::uniform_int_distribution<int> uniform_dist(1, 6);
int mean = uniform_dist(e1);
std::cout << "Randomly-chosen mean: " << mean << '\n';

// Generate a normal distribution around that mean
std::seed_seq seed2{r(), r(), r(), r(), r(), r(), r(), r()};
std::mt19937 e2(seed2);
std::normal_distribution<> normal_dist(mean, 2);

std::map<int, int> hist;
for (int n = 0; n < 10000; ++n) {
    ++hist[std::round(normal_dist(e2))];
}
std::cout << "Normal distribution around " << mean << ":\n";
for (auto p : hist) {
    std::cout << std::fixed << std::setprecision(1)
    << std::setw(2) << p.first << ' ' <<
    std::string(p.second/200, '*') << '\n';
}
}
```

Program Complexity ?

```
HRESULT BasicFileOpen()
{
    // CoCreate the File Open Dialog object.
    IFileDialog *pfd = NULL;
    HRESULT hr = CoCreateInstance(CLSID_FileOpenDialog, NULL, CLSCTX_INPROC_SERVER, IID_PPV_ARGS(&pfd));
    if (SUCCEEDED(hr)) {
        // Create an event handling object, and hook it up to the dialog.
        IFileDialogEvents *pfde = NULL;
        hr = CDialogEventHandler_CreateInstance(IID_PPV_ARGS(&pfde));
        if (SUCCEEDED(hr)) {
            // Hook up the event handler.
            DWORD dwCookie;
            hr = pfd->Advise(pfde, &dwCookie);
            if (SUCCEEDED(hr)) {
                // Set the options on the dialog.
                DWORD dwFlags;

                // Before setting, always get the options first in order
                // not to override existing options.
                hr = pfd->GetOptions(&dwFlags);
                if (SUCCEEDED(hr)) {
                    // In this case, get shell items only for file system items.
                    hr = pfd->SetOptions(dwFlags | FOS_FORCEFILESYSTEM);
                    if (SUCCEEDED(hr)) {
                        // Set the file types to display only.
                        // Notice that this is a 1-based array.
                        hr = pfd->SetFileTypes(ARRAYSIZE(c_rgSaveTypes), c_rgSaveTypes);
                        if (SUCCEEDED(hr)) {
                            // Set the selected file type index to Word Docs for this example.
                            hr = pfd->SetFileTypeIndex(INDEX_WORDDOC);
                            if (SUCCEEDED(hr)) {
                                // Set the default extension to be ".doc" file.
                                hr = pfd->SetDefaultExtension(L".doc;docx");
                                if (SUCCEEDED(hr)) {
                                    // Show the dialog
                                    hr = pfd->Show(NULL);
                                    if (SUCCEEDED(hr)) {
                                        // Obtain the result once the user clicks
                                        // the 'Open' button.
                                        // The result is an IShellItem object.
                                        IShellItem *psiResult;
                                        hr = pfd->GetResult(&psiResult);
                                        if (SUCCEEDED(hr)) {
                                            // We are just going to print out the
                                            // name of the file for sample sake.
                                            PWSTR pszFilePath = NULL;
                                            hr = psiResult->GetDisplayName(SIGDN_FILESYSPATH, &pszFilePath);
                                            if (SUCCEEDED(hr))
                                                TaskDialog(NULL, NULL, L"CommonFileDialogApp", pszFilePath, NULL, TDCBF_OK_BUTTON, TD_INFORMATION_ICON, NULL);
                                            CoTaskMemFree(pszFilePath);
                                        }
                                        psiResult->Release();
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        // Unhook the event handler.
        pfde->Unadvise(dwCookie);
        pfde->Release();
    }
    pfd->Release();
    return hr;
}
```

Reduce Complexity

```
void DoThing(int index)
{
    if (IsValidIndexOfOtherThing(index))
    {
        if (CanDoSomethingWithNumber(index))
        {
            if (CheckSomethingCriticalAboutValue(index))
            {
                for (auto const& value : GetData(index))
                {
                    switch (value % 3)
                    {
                        case 0:
                            PrintFoo(value);
                            break;

                        case 1:
                            PrintBar(value);
                            break;

                        case 2:
                            PrintBaz(value);
                            break;
                    }
                }
            }
        }
    }
}
```



```
void DoThing(int index)
{
    if (!IsValidIndexOfOtherThing(index))
    {
        return;
    }

    if (!CanDoSomethingWithNumber(index))
    {
        return;
    }

    if (!CheckSomethingVeryCriticalAboutValue(index))
    {
        return;
    }

    for (auto const& value : GetValuesSimilarTo(index))
    {
        ProcessValue(value);
    }
}
```

Flatten, using guards



Guard Pattern

```
/// e.g., "my_key: 123"
pub fn key_num<'a>(item: &'a str) → Result<(&'a str, i32) > {
    if let Some((key, val)) = item split_once(':') {
        if let Ok(val) = val.trim().parse::<i32>() {
            → Ok((key, val))
        } else {
            Err(Error::Static("Can't parse integer"))
        }
    } else {
        Err(Error::Static("Invalid format"))
    }
}
```



Guard Pattern

```
/// e.g., "my_key: 123"
pub fn key_num<'a>(item: &'a str) → Result<(&'a str, i32) > {
    let Some((key, val)) = item split_once(':') else {
        return Err(Error::Static("Invalid format"));
    };

    let Ok(val) = val.trim().parse::<i32>() else {
        return Err(Error::Static("Can't parse integer"));
    };

    → Ok((key, val))
}
```



Guard Pattern

```
func getMeaningOfLife() -> Int? {  
    42  
}  
  
func printMeaningOfLife() {  
    if let name = getMeaningOfLife() {  
        print(name)  
    }  
}
```

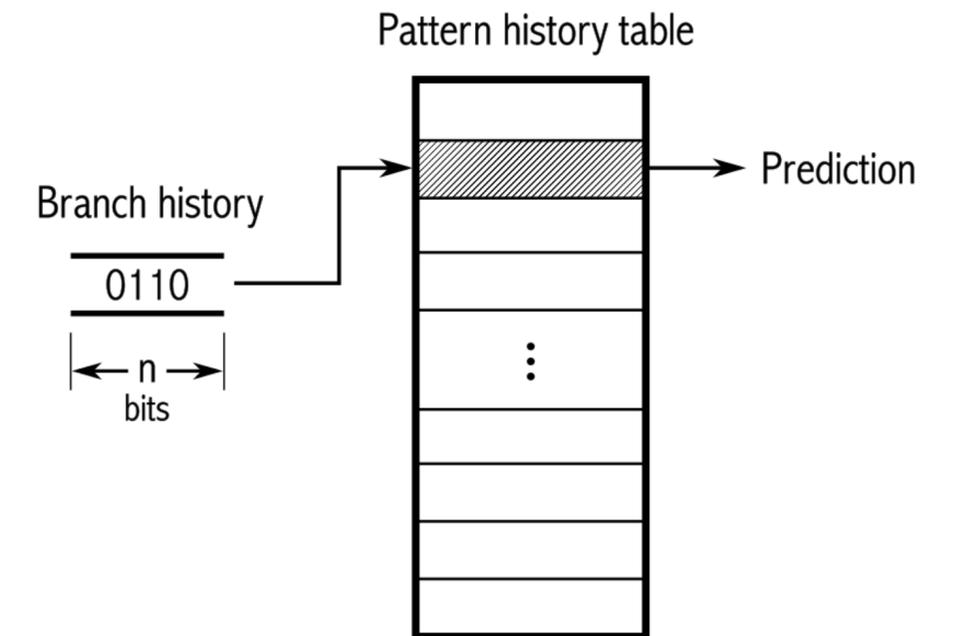


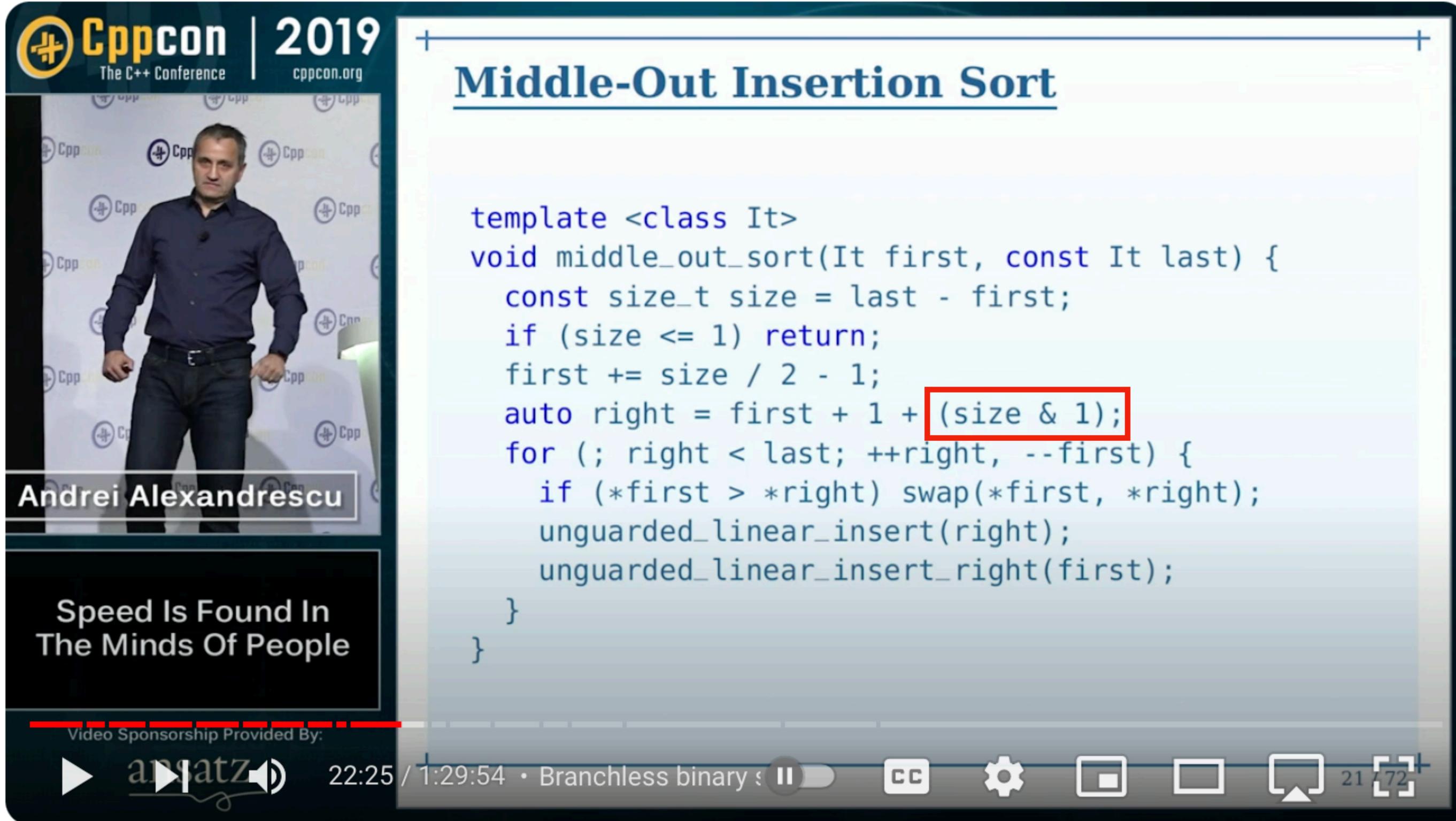
Guard Pattern

```
func printMeaningOfLife() {  
    guard let name = getMeaningOfLife() else {  
        return  
    }  
  
    print(name)  
}
```



Code that is left-leaning is fast





Cppcon | **2019**
The C++ Conference | cppcon.org

Andrei Alexandrescu

Speed Is Found In
The Minds Of People

Video Sponsorship Provided By: **arsatz**

22:25 / 1:29:54 • Branchless binary

Middle-Out Insertion Sort

```
template <class It>
void middle_out_sort(It first, const It last) {
    const size_t size = last - first;
    if (size <= 1) return;
    first += size / 2 - 1;
    auto right = first + 1 + (size & 1);
    for (; right < last; ++right, --first) {
        if (*first > *right) swap(*first, *right);
        unguarded_linear_insert(right);
        unguarded_linear_insert_right(first);
    }
}
```

“Code that is left-leaning is fast”

- Andrei Alexandrescu

```
auto right = first + 1 + (size & 1);
```

```
 if (size & 1) right++;
```

Position in the middle of the array - but differently **if** we have odd or even number of elements.

But there is no **`if`** statement!

Integrating the **conditional** within the arithmetic, to **avoid branching**. (no jumps!)

Incidental vs. deliberate symmetry

Should We Care?

We should be looking to identify patterns in code, to see when such constructs exhibit some sort of symmetry that is advantageous in some way for:

- reliability
- performance
- maintenance/extensibility
- discoverability

Incrementing variables in for-loops:

`i++`

- overused
- nonsensical
- imbalanced

`i--=-1`

- hipster
- expressive
- **symmetric**

credit: *probably* Ólafur Waage

Symmetry in Code Should We Care?

ACCU

April 2024

 @ciura_victor

 @ciura_victor@hachyderm.io

 @ciuravictor.bsky.social

Victor Ciura
Principal Engineer
M365 Substrate

