

**accu**  
conFerence  
**2024**

# Symmetry in Code

## Should We Care?

**Victor Ciura**

# Symmetry in Code Should We Care?

ACCU

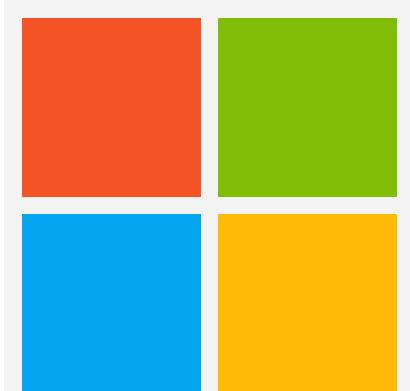
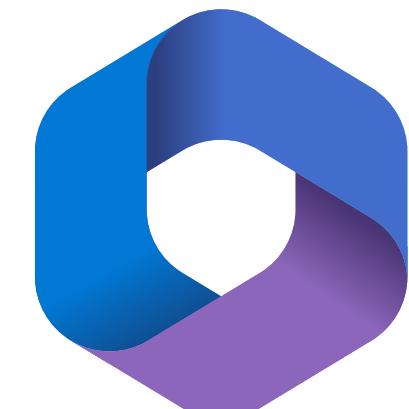
April 2024

 @ciura\_victor

 @ciura\_victor@hachyderm.io

 @ciuravictor.bsky.social

**Victor Ciura**  
Principal Engineer  
M365 Substrate



# Abstract

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.

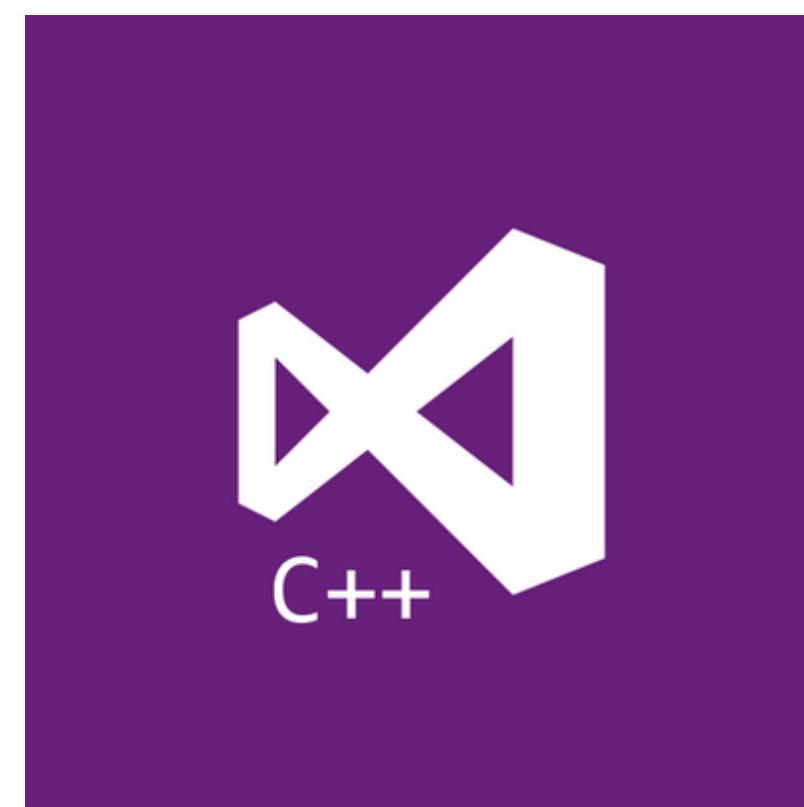
# About me



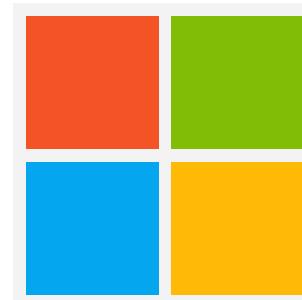
**Advanced Installer**



**Clang Power Tools**



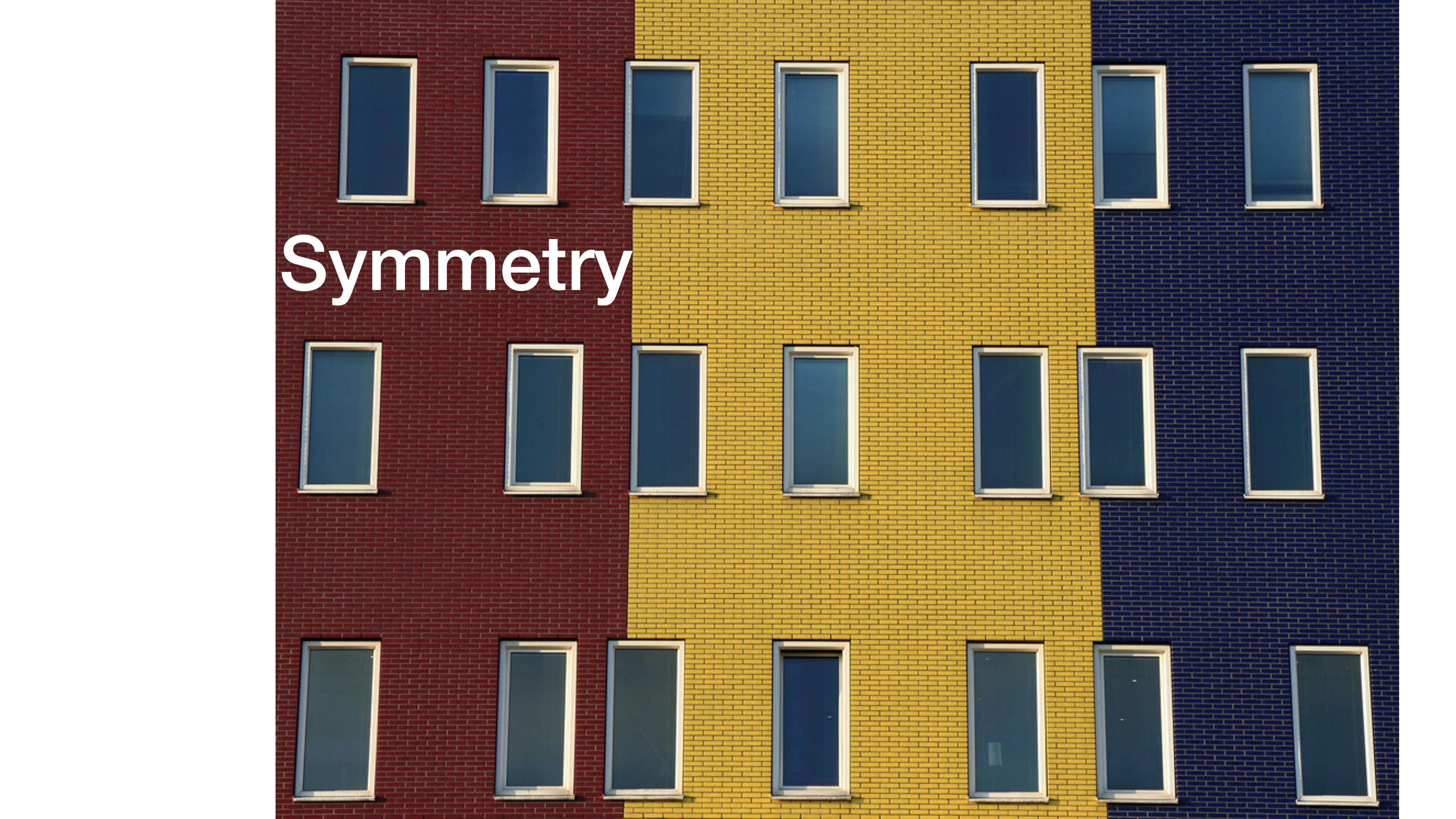
**Visual C++**



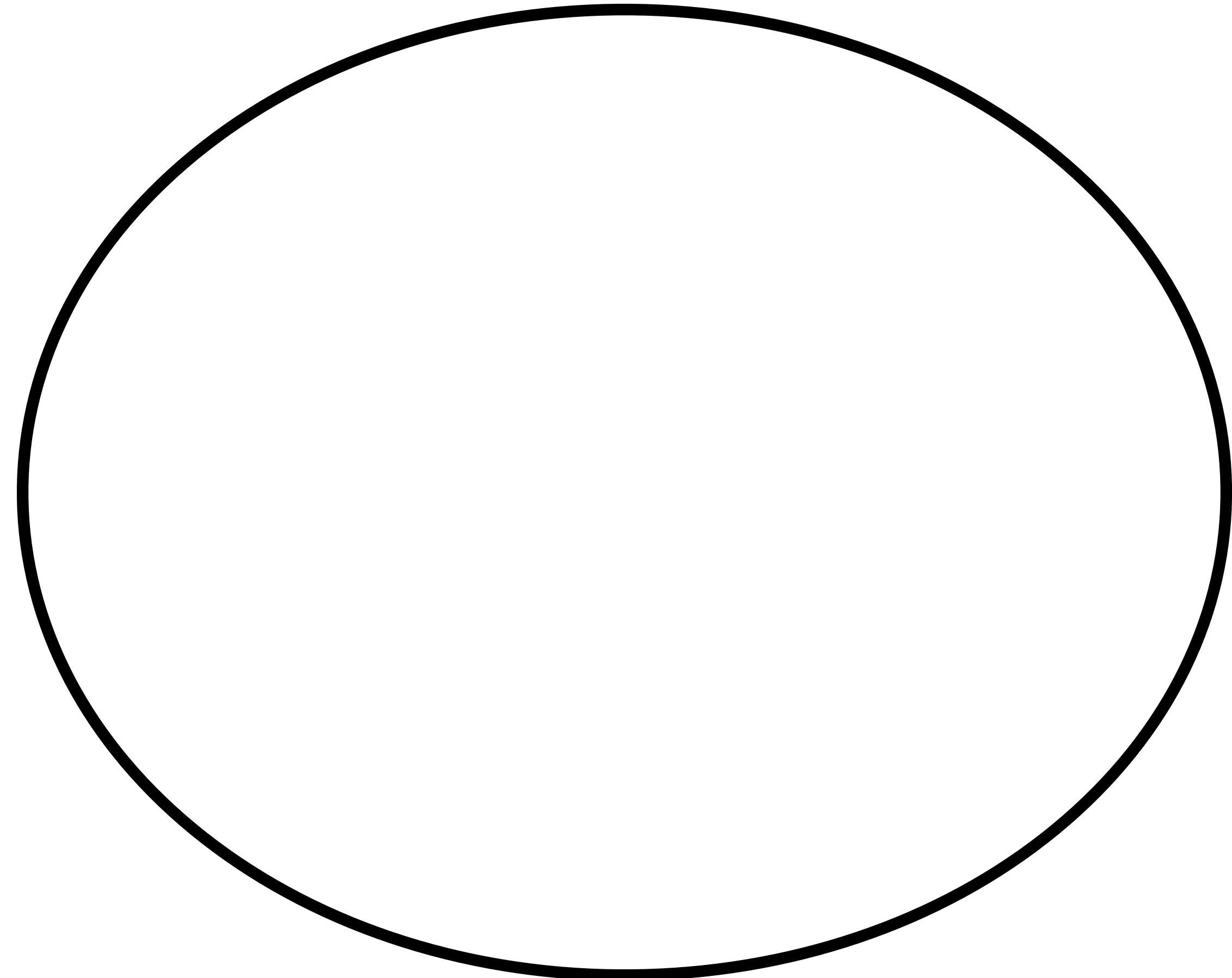
**M365 Substrate**

-  [@ciura\\_victor](https://twitter.com/ciura_victor)
-  [@ciura\\_victor@hachyderm.io](mailto:@ciura_victor@hachyderm.io)
-  [@ciuravictor.bsky.social](https://ciuravictor.bsky.social)



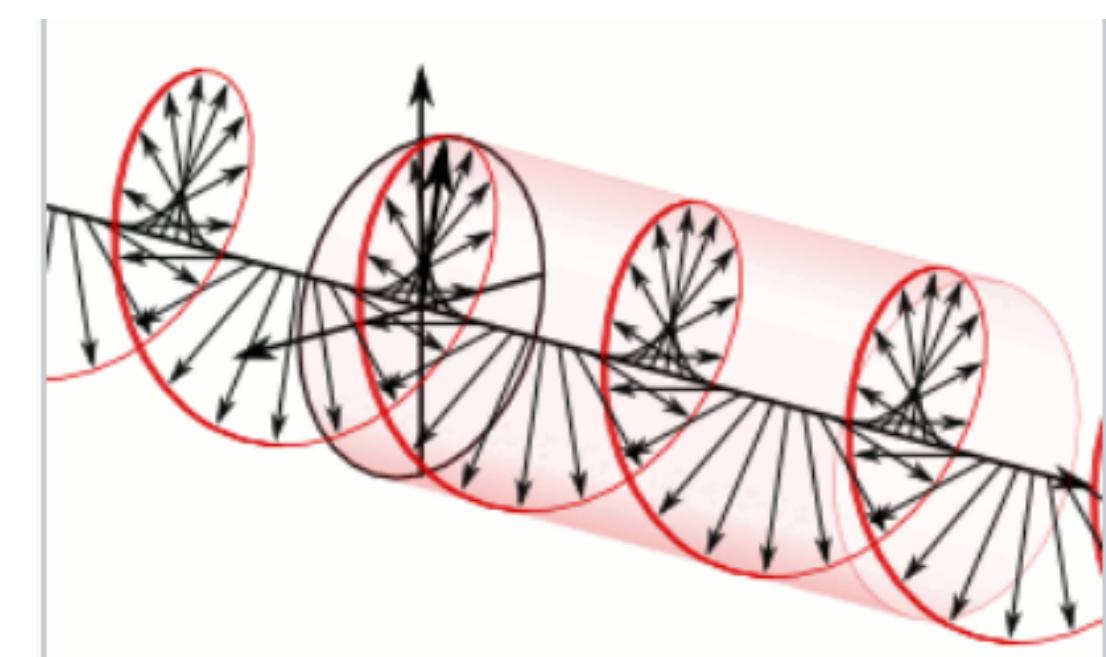
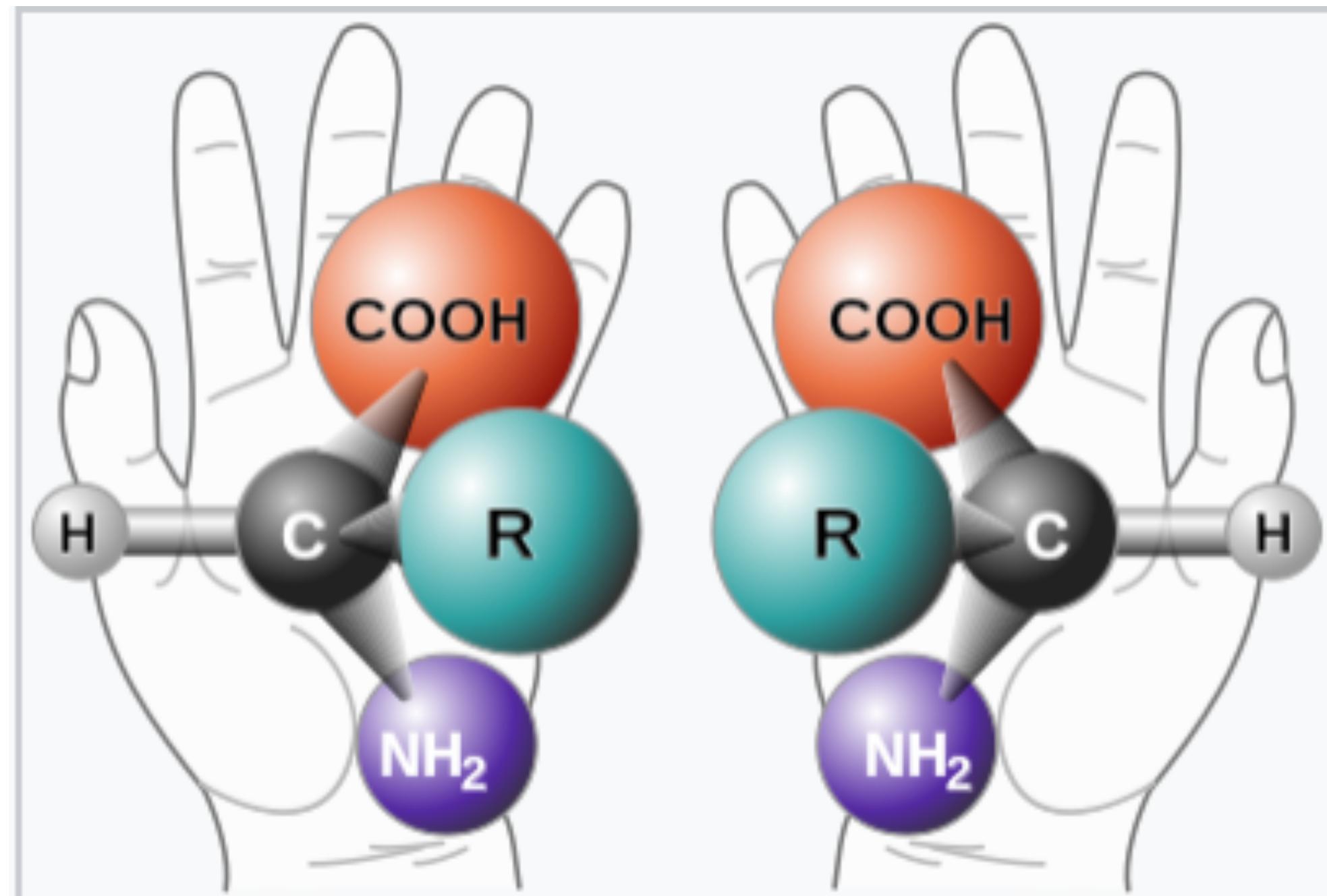


# Symmetry

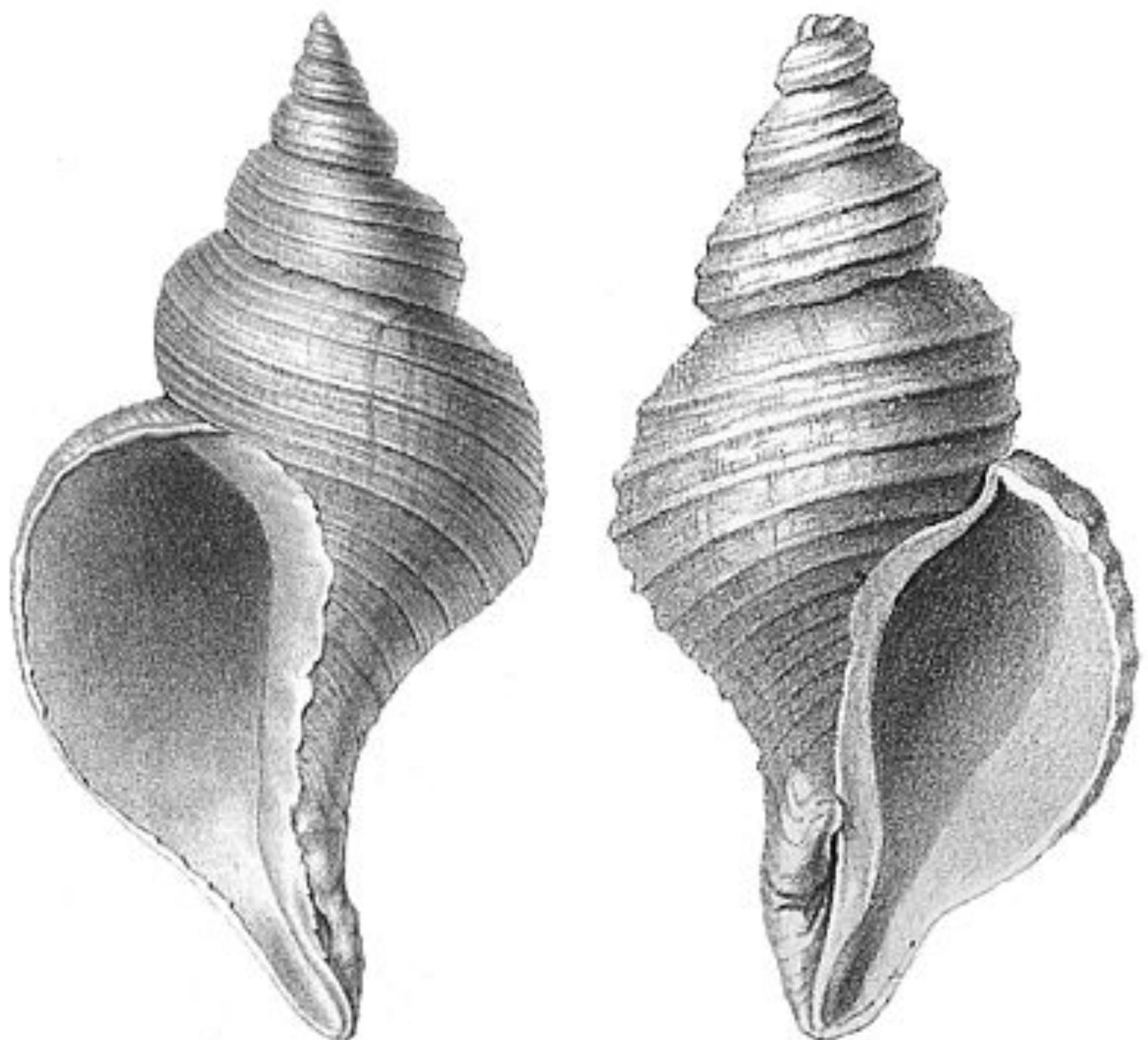


In fact, it is like the old idea of ancient GREEKS that circles were perfect, and it was rather horrible for them to believe that the planetary orbits were not circles, but only nearly circles.

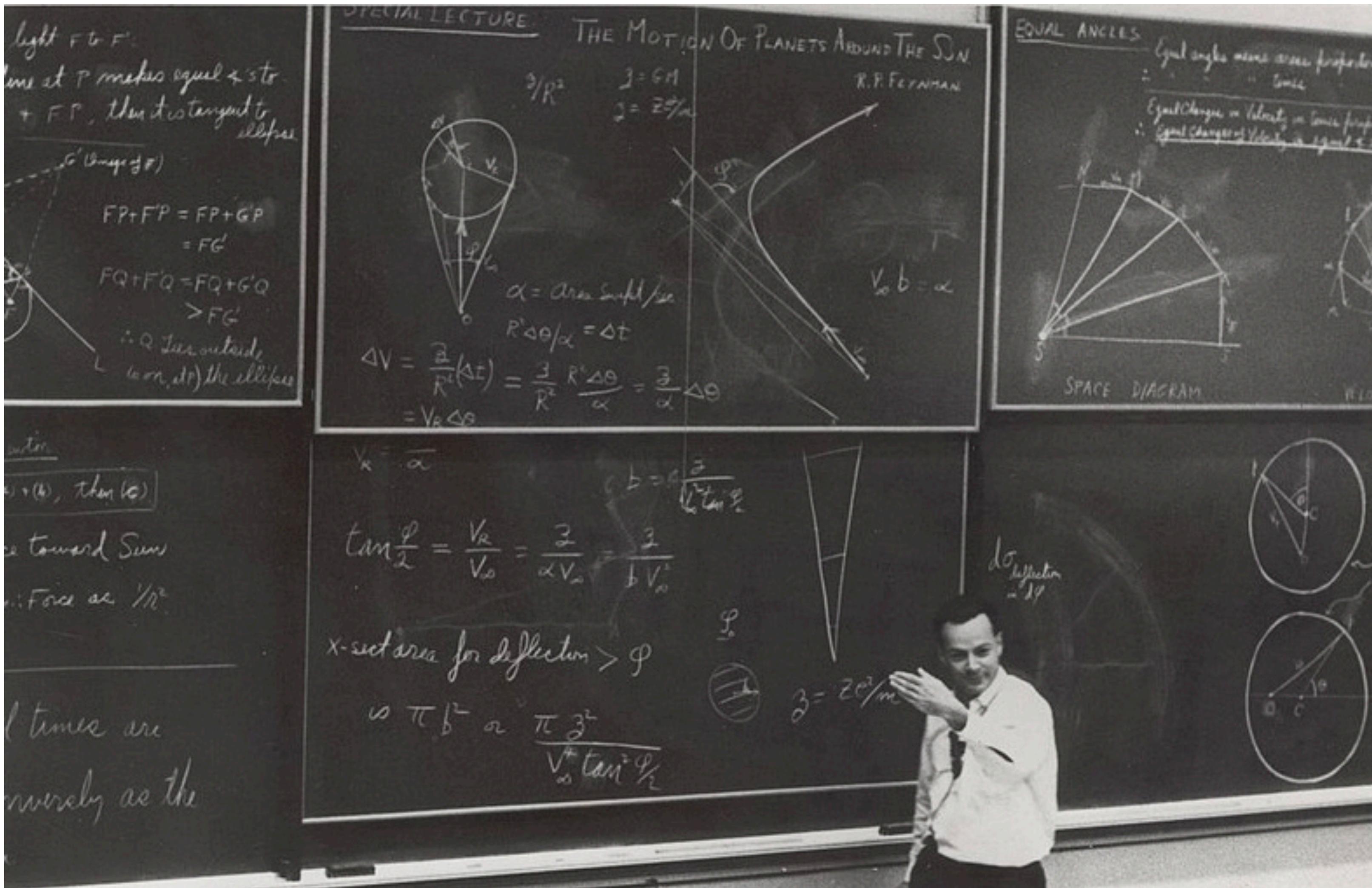
# Chirality



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://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 "looks" the same

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



[ted.com/talks/marcus\\_du\\_sautoy\\_symmetry\\_reality\\_s\\_riddle](https://ted.com/talks/marcus_du_sautoy_symmetry_reality_s_riddle)



## Beauty, truth and ... physics?

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

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

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

[ted.com/talks/murray\\_gell\\_mann](https://ted.com/talks/murray_gell_mann)

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

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

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

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

---

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

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

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

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

Original form

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

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

Simplified using rotational symmetry

Further simplified using the symmetry of special relativity

# The Shape of A Program

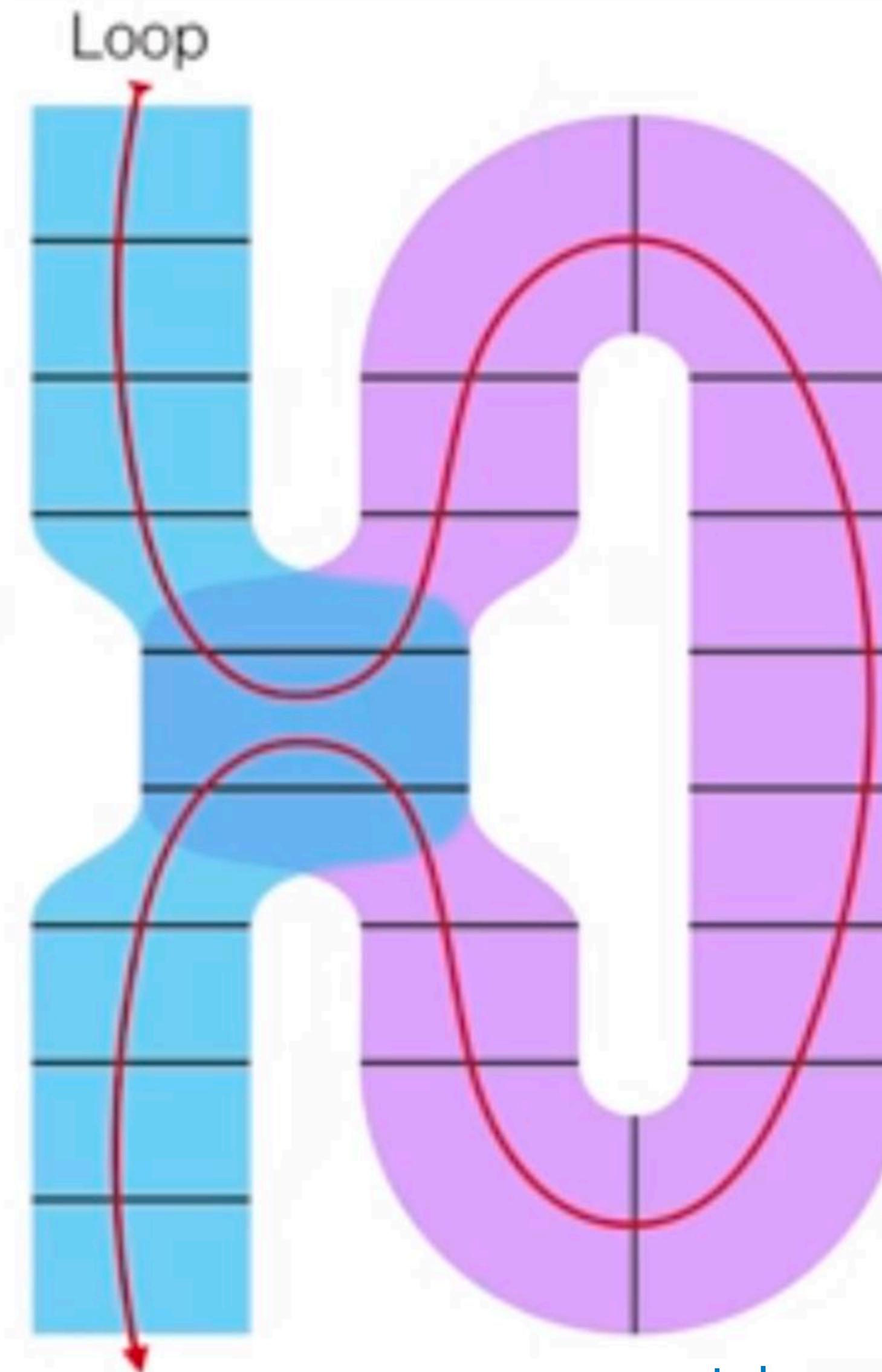
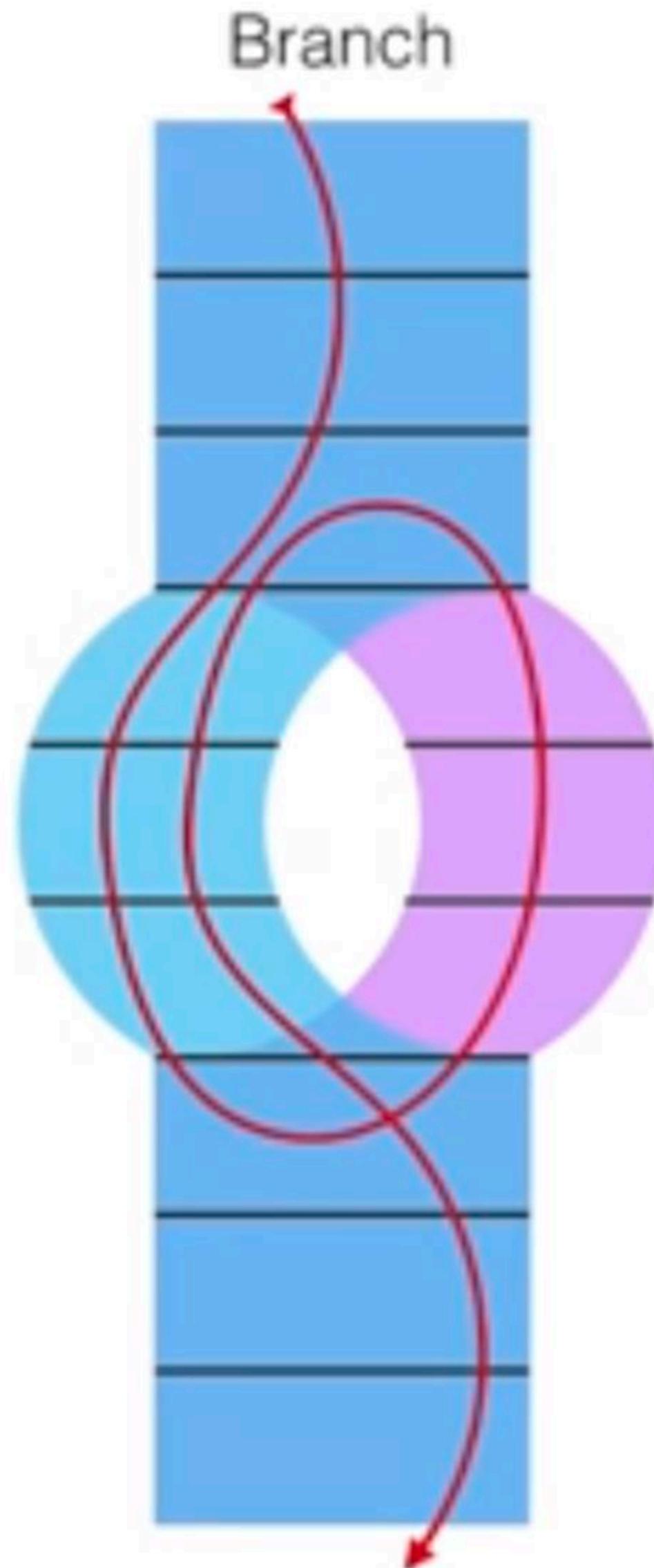
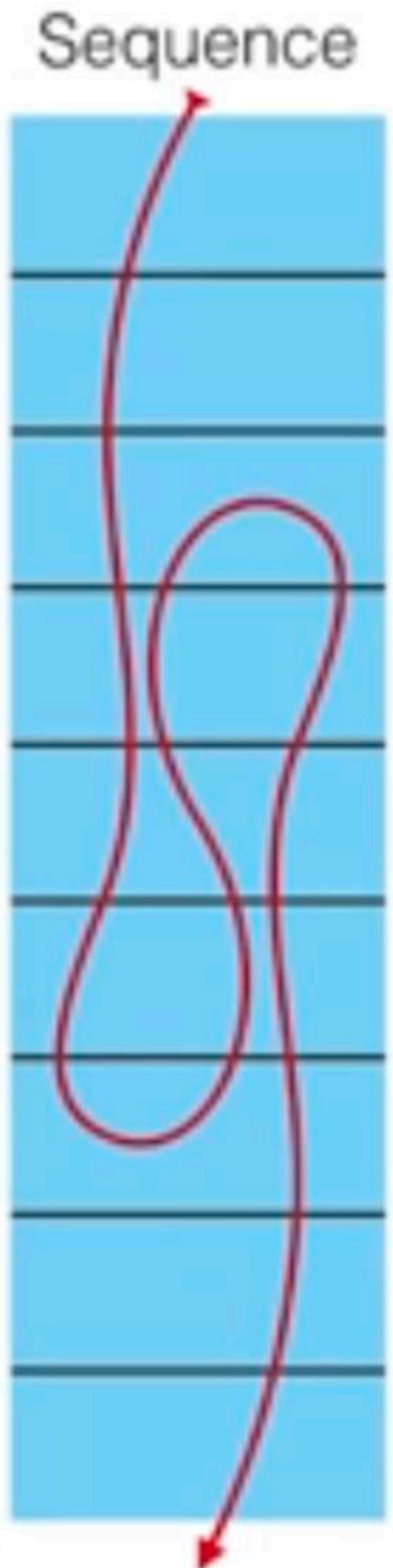
The screenshot shows a video player interface. On the left, a white rectangular box contains a diagram of two overlapping circles, one blue labeled 'e' and one purple labeled 'f'. They overlap in the center, and their boundaries define a shape. Arrows point from the bottom of each circle to a horizontal line at the bottom of the slide. To the right of the diagram is a text box containing the following text:

If we expand Ⓜ and Ⓛ to share entrances and exits, they remain alternative possibilities.

On the right side of the video player, there is a video frame showing a woman with long blonde hair and glasses, wearing a blue patterned top, speaking. Below the video frame, the text "Lisa Lippincott" is displayed. Underneath her name, the title "The Shape of a Program" is shown. At the bottom of the video frame, the text "Video Sponsorship Provided By: JET BRAINS" is visible. The video player has a dark background with a green banner at the top that reads "C++ now 2018 MAY 7 - 11 cppnow.org". The bottom of the video player shows standard controls: play/pause, volume, time (0:00 / 1:15:07), and other playback options.

[youtube.com/watch?v=QFIOE1jKv30](https://youtube.com/watch?v=QFIOE1jKv30)

# The Shape of A Program



[youtube.com/watch?v=QFIOE1jKv30](https://youtube.com/watch?v=QFIOE1jKv30)

# Shapes of Code



The saw

[fluentcpp.com/2020/01/14/the-shapes-of-code/](https://fluentcpp.com/2020/01/14/the-shapes-of-code/)

# Shapes of Code



## The paragraphs

[fluentcpp.com/2020/01/14/the-shapes-of-code/](https://fluentcpp.com/2020/01/14/the-shapes-of-code/)

# Shapes of Code



**The paragraphs with headers**

[fluentcpp.com/2020/01/14/the-shapes-of-code/](https://fluentcpp.com/2020/01/14/the-shapes-of-code/)

# Shapes of Code

**if**   
  
**else** 

**if**   
  
**else** 

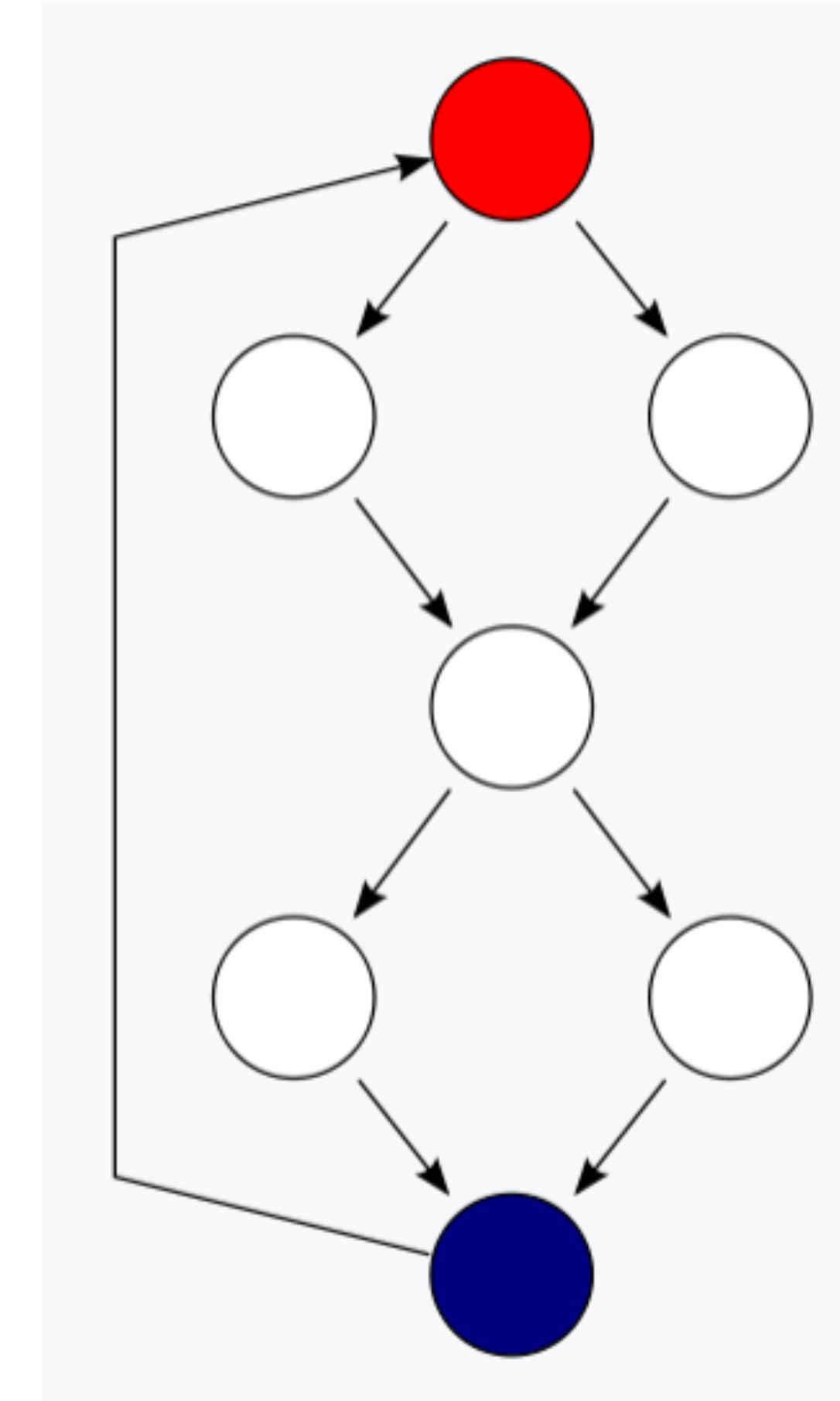
**The unbalanced `if` blocks**

[fluentcpp.com/2020/01/14/the-shapes-of-code/](https://fluentcpp.com/2020/01/14/the-shapes-of-code/)

# Cyclomatic Complexity

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

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



[wikipedia.org/wiki/Cyclomatic\\_complexity](https://wikipedia.org/wiki/Cyclomatic_complexity)

# The Shape of A Program

The screenshot shows a video player interface. At the top right, the text "cppcon | 2018" and "THE C++ CONFERENCE • BELLEVUE, WASHINGTON" is displayed. The main content area features a large white title "THE SHAPE OF A PROGRAM" on a black background. To the right of the title, a video frame shows a man with short brown hair, wearing a dark suit jacket over a light blue shirt, speaking into a microphone. Below the video frame, the name "JAMES McNELLIS" is displayed in a white box. To the right of the video frame, the title "The Shape of a Program" is shown in white text on a black background. At the bottom of the screen, there is a navigation bar with icons for play, volume, and other controls, along with the text "0:07 / 5:06 • Lisa Lippincott >".

[youtube.com/watch?v=P2IxGnbDkDI](https://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 *pfld = NULL;
    HRESULT hr = CoCreateInstance(CLSID_FileOpenDialog, NULL, CLSCTX_INPROC_SERVER, IID_PPV_ARGS(&pfld));
    if (SUCCEEDED(hr)) {
        // Create an event handling object, and hook it up to the dialog.
        IFileDialogEvents *pfde = NULL;
        hr = CDialEventHandle_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;

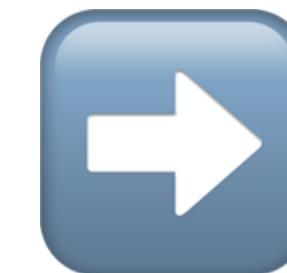
                // Before setting, always get the options first in order
                // not to override existing options.
                hr = pfd->GetOptions(&dwFlags);
                if (SUCCEEDED(hr)) {
                    // In this case, get shell items only for file system items.
                    hr = 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, TD_CBF_OK_BUTTON, TD_INFORMATION_ICON, NULL);
                                                CoTaskMemFree(pszFilePath);
                                            }
                                            psiResult->Release();
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    // Unhook the event handler.
    pfd->Unadvise(dwCookie);
}
pfde->Release();
pfld->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 getMeaning0fLife() -> Int? {  
    42  
}  
  
func printMeaning0fLife() {  
    if let name = getMeaning0fLife() {  
        print(name)  
    }  
}
```



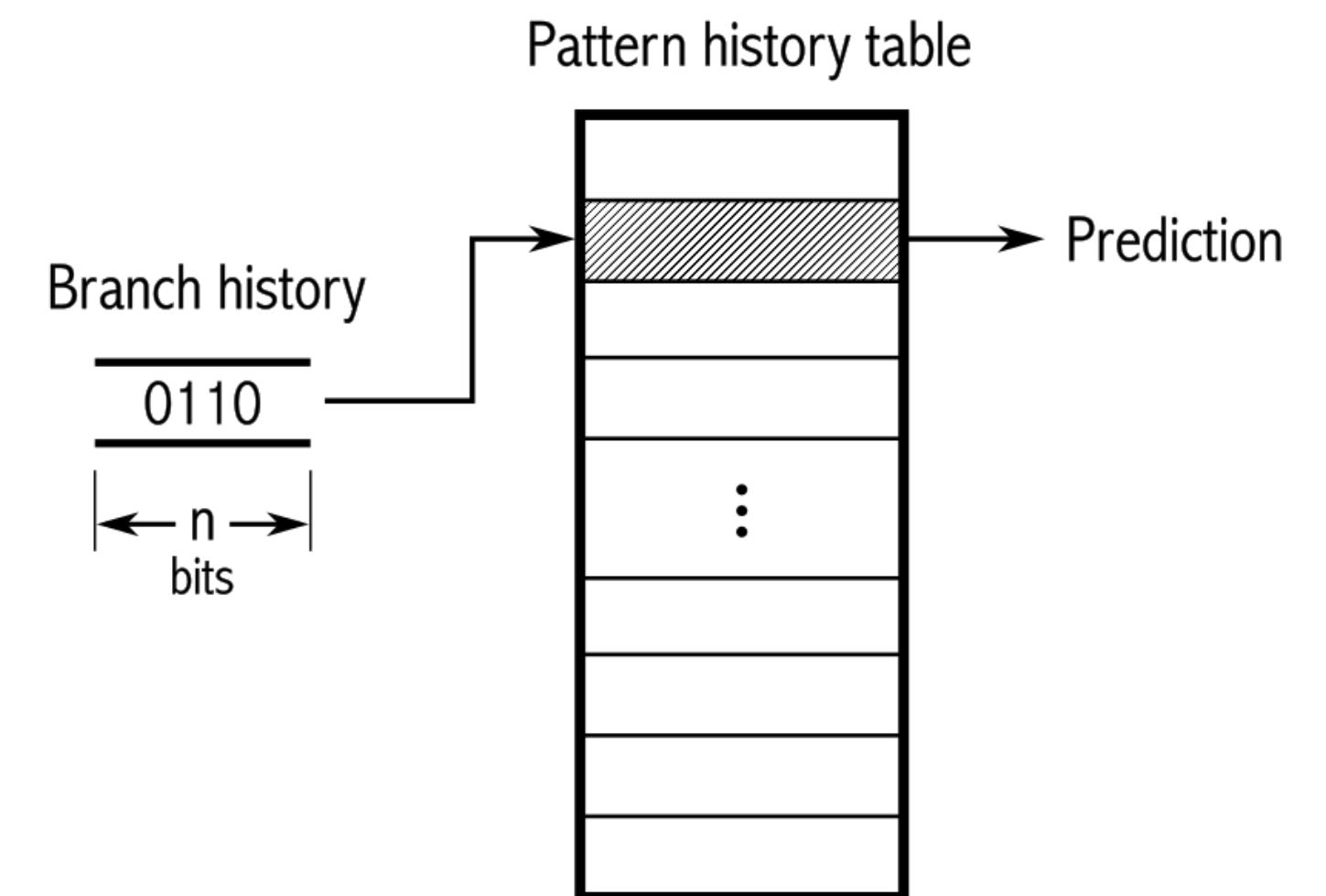
# Guard Pattern

```
func printMeaning0fLife() {  
    guard let name = getMeaning0fLife() else {  
        return  
    }  
  
    print(name)  
}
```





## Code that is left-leaning is fast



The video player interface shows a speaker, Andrei Alexandrescu, standing on stage. The background is a repeating Cppcon logo. A subtitle box identifies him as "Andrei Alexandrescu". Below the video, a dark overlay contains the text "Speed Is Found In The Minds Of People". At the bottom, a red bar indicates "Video Sponsorship Provided By: ansatz". The video progress bar shows 22:25 / 1:29:54, and the title "Branchless binary search".

## 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);
    }
}
```

[youtube.com/watch?v=FJJTYQYB1JQ](https://youtube.com/watch?v=FJJTYQYB1JQ)



**“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:

- ➊ reliability
- ➋ performance
- ➌ maintenance/extensibility
- ➍ 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

