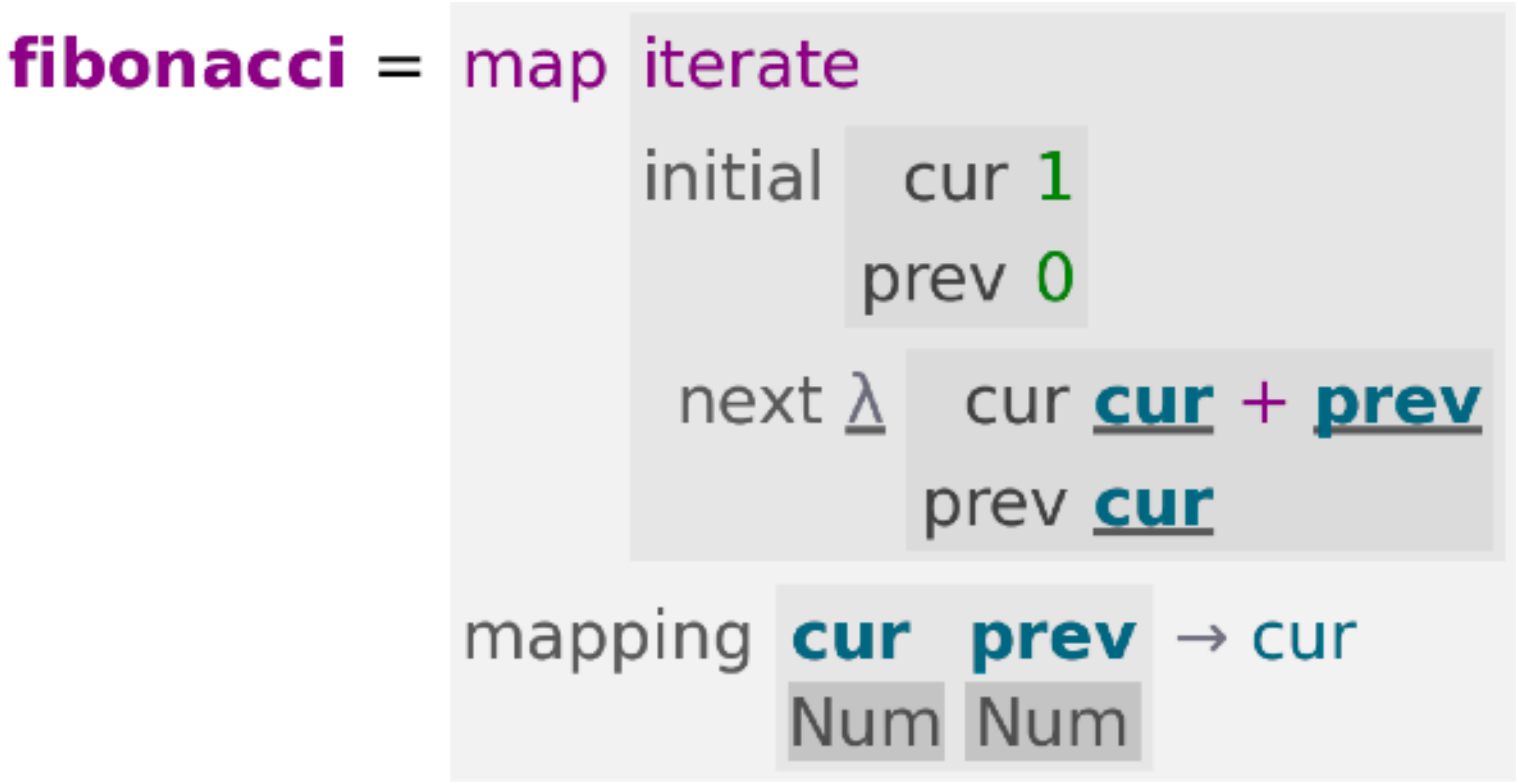
# Projectional Syntactic Sugar



```
fibonacci = map iterate
                  initial   cur 1
                           prev 0
                    next λ    cur cur + prev
                             prev cur
            mapping cur  prev → cur
                    Num  Num
```

Lamdu displays the “next" function using *light lambda* syntax while the “mapping” function is displayed with plain lambda syntax

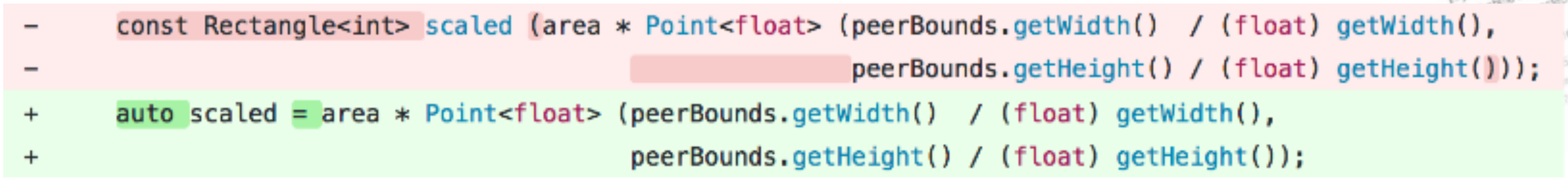
Lamdu automatically presents code with syntactic sugars

go fmt

# Manual Formatting

Programmers maintain whitespace, deciding how to indent their code, split their lines and align function arguments, to make the code readable while fitting the screen width.

Developers Who Use Spaces Make More Money Than Those Who Use Tabs

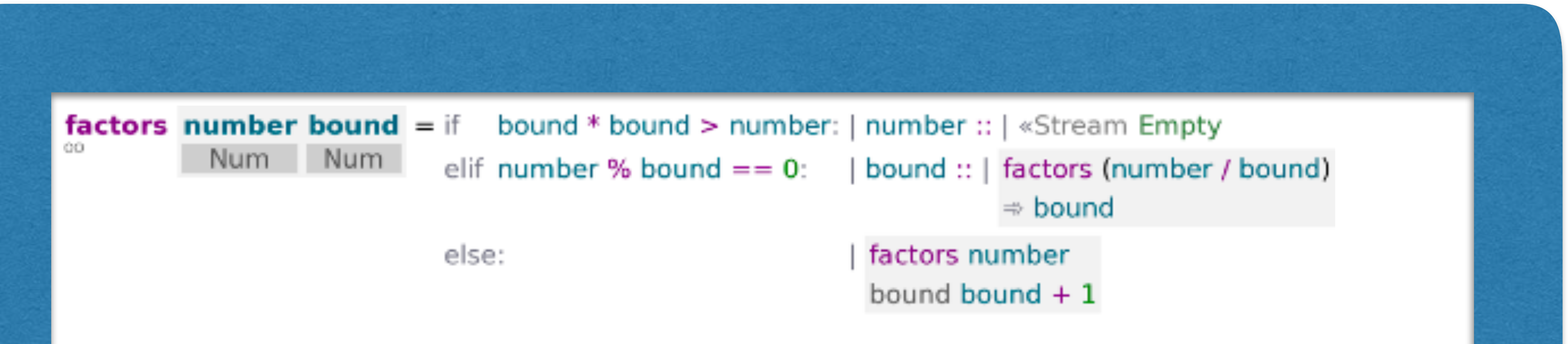


```
-     const Rectangle<int> scaled (area * Point<float> (peerBounds.getWidth()  / (float) getWidth(),
-                                                       peerBounds.getHeight() / (float) getHeight()));
+     auto scaled = area * Point<float> (peerBounds.getWidth()  / (float) getWidth(),
+                                        peerBounds.getHeight() / (float) getHeight());
```

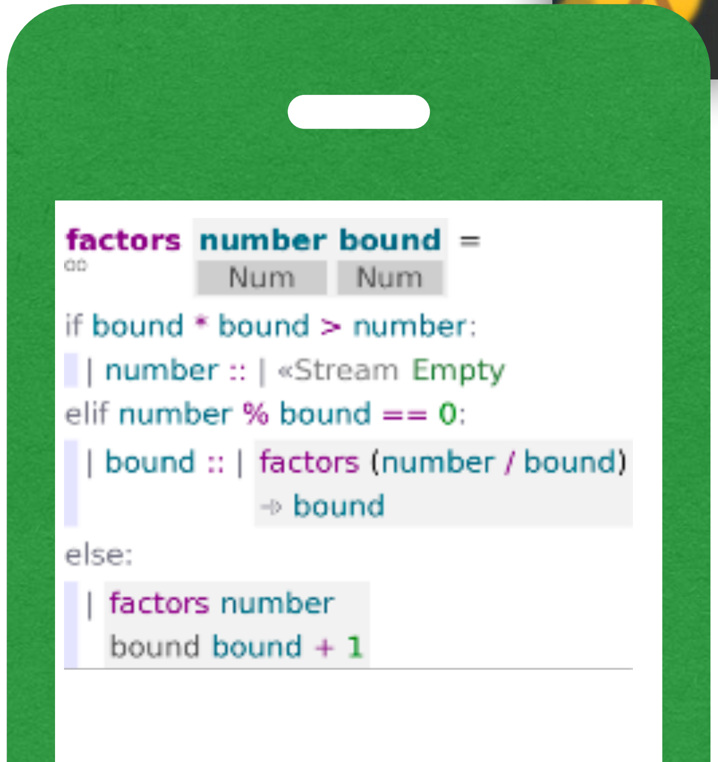
A typical C++ code diff. The programmer maintains the spacing manually.

## Automatic Layout

- Convenient
- Responsive
- Consistent
- No conflicts



```
factors number bound = if    bound * bound > number:  | number :: | «Stream Empty
        Num    Num      elif  number % bound == 0:     | bound :: | factors (number / bound)
                                                                   ⇒ bound
                        else:                          | factors number
                                                         bound bound + 1
```

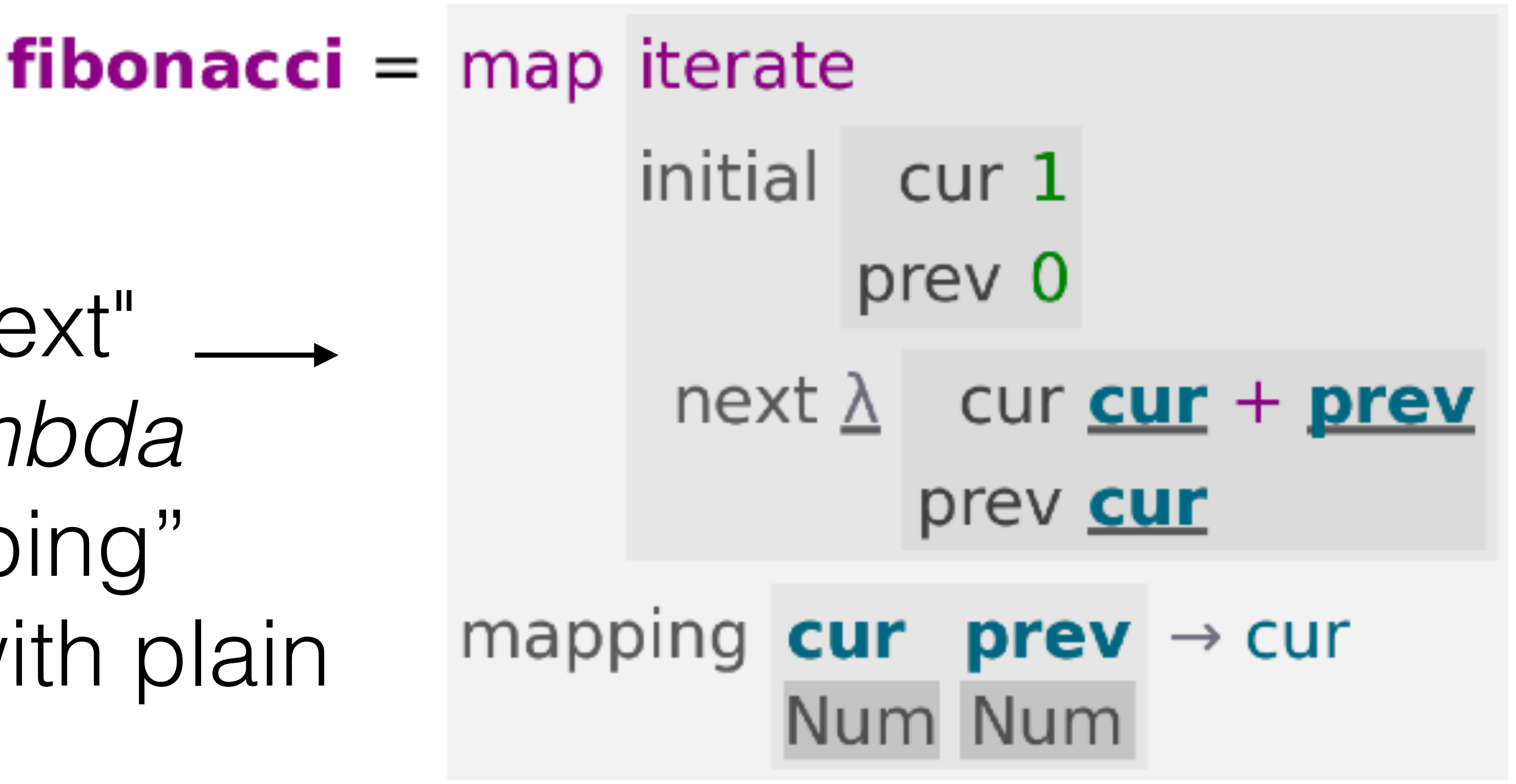


```
factors number bound =
        Num    Num
if bound * bound > number:
  | number :: | «Stream Empty
elif number % bound == 0:
  | bound :: | factors (number / bound)
               ⇒ bound
else:
  | factors number
    bound bound + 1
```

# Projectional Syntactic Sugar

Lamdu automatically presents code with syntactic sugars

Lamdu displays the "next" function using *light lambda* syntax while the "mapping" function is displayed with plain lambda syntax



```
fibonacci = map iterate
                  initial  cur 1
                           prev 0
                   next λ  cur cur + prev
                           prev cur
            mapping cur  prev → cur
                    Num  Num
```