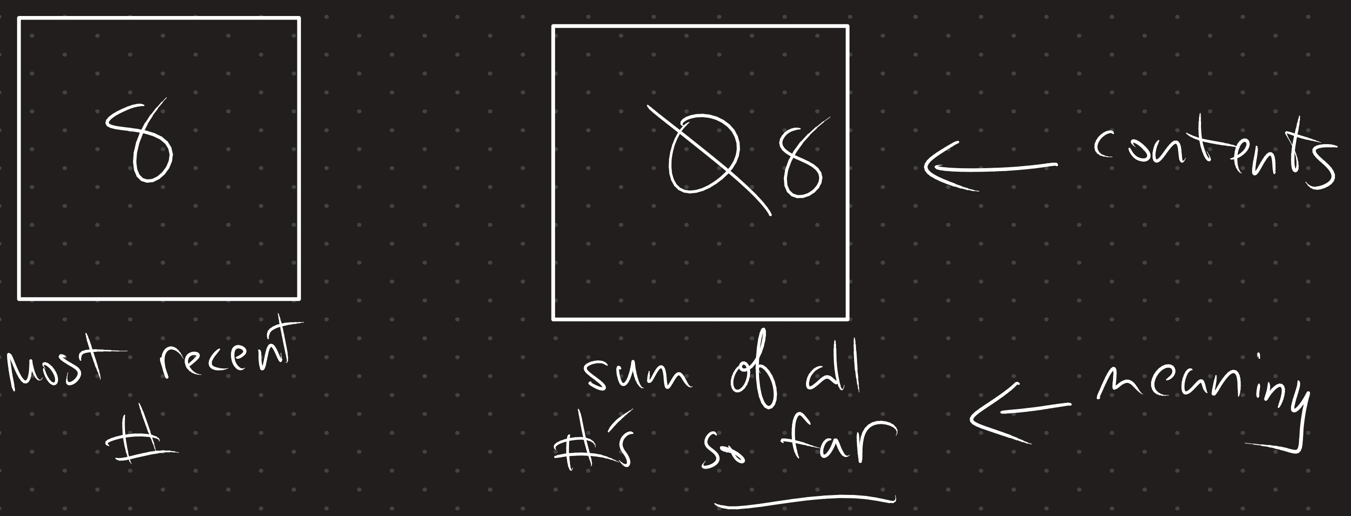
From last time: computing the sum of integers given on stdin (cin).



① write 0 to one variable
② listen for new # on other variable
→ ③ add both, write the result to the sum so far.
④ Repeat from ② until there are no more #'s

Observation: each of our variables has a well-defined meaning.

This idea is usually refered to as an "invariant".

The sum program might have reminded you of the largest # program, and it should! Both are special cases of a pattern called folding.

Setup: given list of values $x_1, x_2, \ldots, x_n$, want to compute

$$x_1 \boxed{?} x_2 \boxed{?} \cdots \boxed{?} x_n$$

for some binary operation $\boxed{?}$.
$\boxed{?}$ could be $+$, $*$, max, min ...

Can always use this meta-solution:

```
int s = e;
int x;
while (cin >> x) {
    s = s [?] x;
}
cout << s << "\n";
```

e must be the neutral element for $\boxed{?}$. That is, for any x,

$$x \boxed{?} e = x$$
$$e \boxed{?} x = x$$

E.g. for $+$, $e = 0$

for $*$, $e = 1$

for max, $e = -\infty$ (INT_MIN for integers)

for min, $e = \infty$ (INT_MAX for ints)