PERSONAL - Finding Loop Invariants

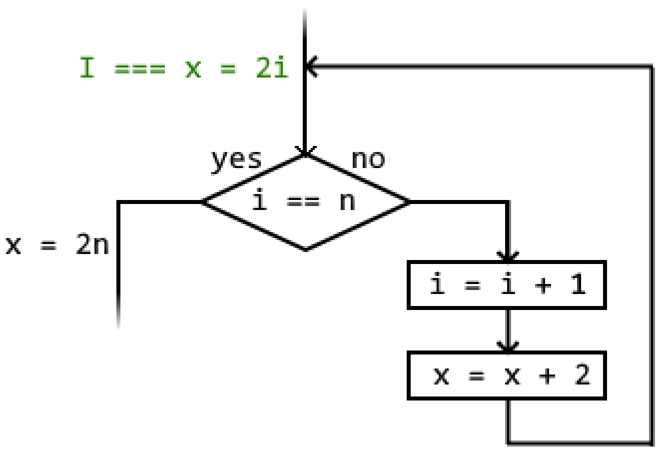
Tips for finding Loop Invariants

TL;DR: Short Summary

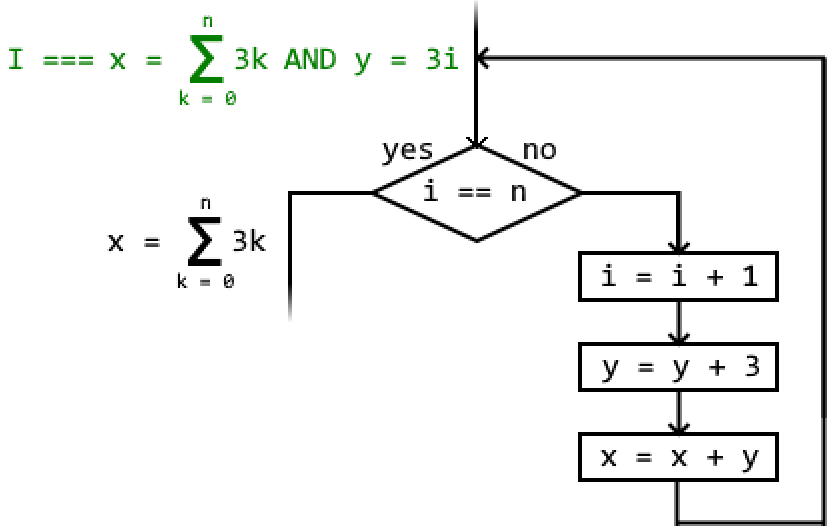
- **Tip 1**: invariant must contain **variable we operate on** and must be as **precise** as possible
- **Tip 2**: **"loop-carried" variables** must be included in the loop invariant
- **Tip 3**: create **generalized tables** to visualize the connections between variables inside the loop
- **Tip 4**: there must be a relation between the variables needed to calculate x **in the loop** and **after the loop**
- **Tip 5**: if the loop condition contains **inequality**, the counter should be "**limited** on the opposite side" in order to reach **equality**
- **Tip 6**: if certain program inputs are **restricted**, these restrictions about input variables should be included in the loop invariant
- **Tip 7**: if the variable in our loop invariant **depends on some other value**, it needs to be included in some way in our loop invariant
- **Tip 8**: when proving termination, r **must be included** in the loop invariant
- **Tip 8.5**: when proving termination, **all variables required** for calculating r must be included in the loop invariant

Detailed Tips

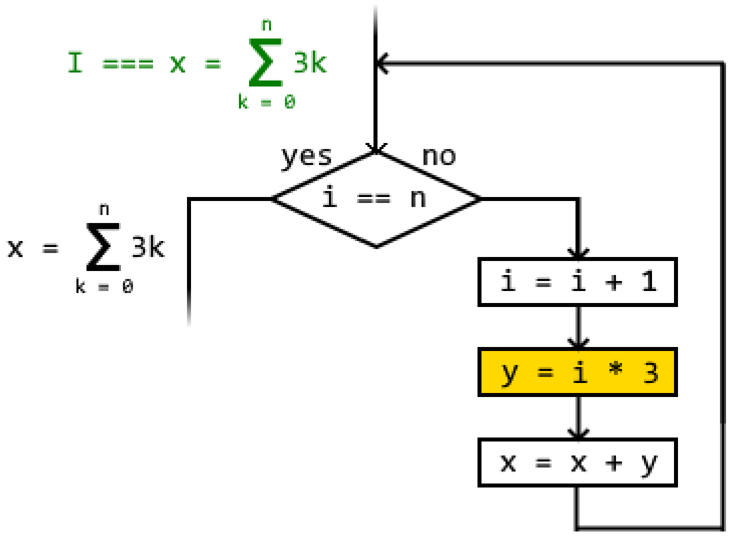
- **Tip 1**: invariant must contain **variable we operate on** and must be as **precise** as possible
  - *example*:
    - $i \geq 0$ would be a **bad invariant**, since we cannot imply $x = 2n$
    - similarly, $x \geq 0$ would also be a **bad invariant**, since we still cannot imply $x = 2n$
    - $x = 2i$, however, would **suffice**



- **Tip 2**: **"loop-carried" variables** (*variables which depend on their value from the previous loop iteration*) must be included in the loop invariant
  - *example*:
    - $x = \sum_{k=0}^{i} 3k$ would be a **bad invariant**, since we know nothing about y
    - $x = \sum_{k=0}^{i} 3k \wedge y = 3i$ would **work**, however



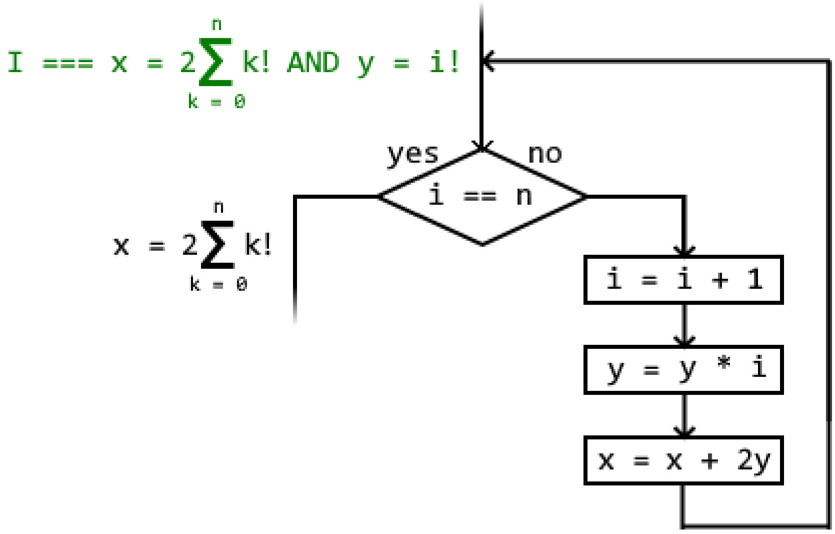
    - if, instead, y was not dependent on its previous value, $x = \sum_{k=0}^{i} 3k$ would suffice



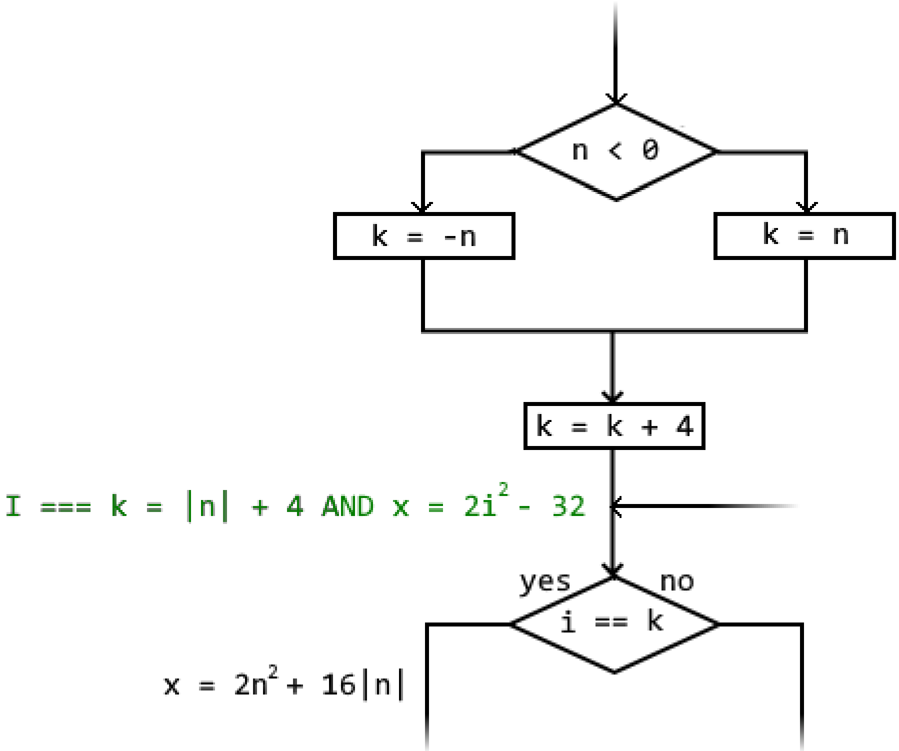
- **Tip 3**: create **generalized tables** to visualize the connections between variables inside the loop
  - *example*:
    - we see that $y = i!$ and $x = 2 \sum_{k=0}^{i} k!$

#	0	1	2	...	i
i	0	1	2	...	i
x	2	2 + 2 * 1	2 + 2 * 1 + 2 * 1 * 2	...	2 + 2 * 1 + 2 * 1 * 2 + ... + 2 * 1 * ... * i
y	1	1	1 * 2	...	1 * 2 * ... * i

- as such, a good invariant would be the **conjunction of these two**, namely $x = 2\sum_{k=0}^{i} k! \wedge y = i!$

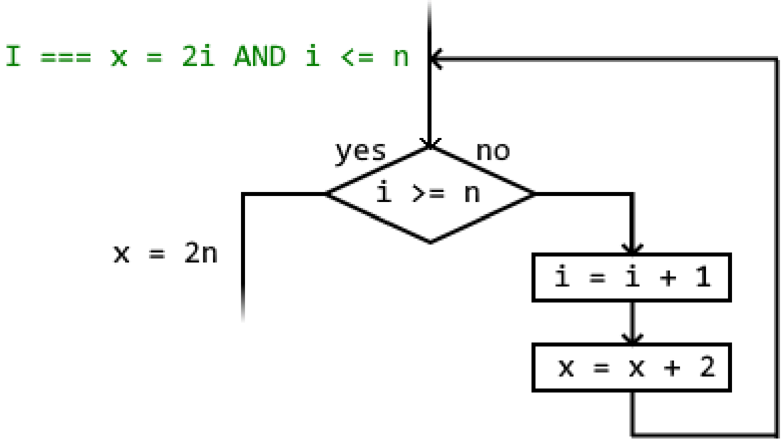


- **Tip 4**: there must be a relation between the variables needed to calculate x **in the loop** and **after the loop**
  - *example*:
    - $x = 2i^2 - 32$ cannot imply $2n^2 + 16|n|$; there needs to be some **connection** between n and i or k
    - we see that $k = |n| + 4$, and incorporating this into our loop invariant would allow us to make the needed connection ($I \equiv k = |n| + 4 \wedge x = 2i^2 - 32$)

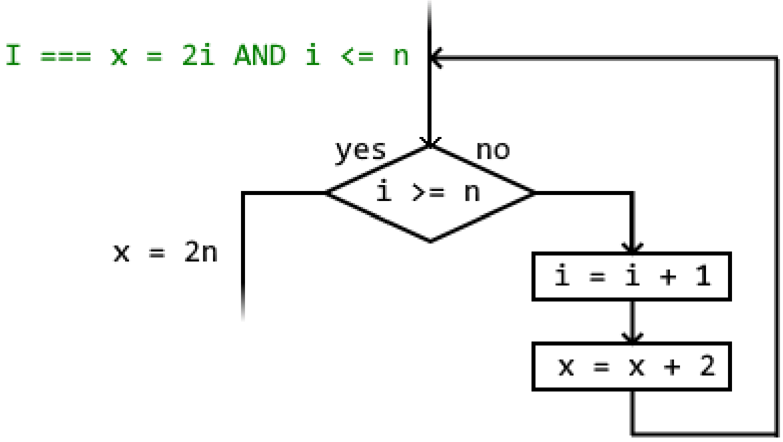


- **Tip 5**: if the loop condition contains **inequality**, the counter should be "**limited** on the opposite side" in order to reach **equality** (e.g. if i is decremented, include $i \geq 0$, else if i is incremented, include $i \leq n$)
  - *example*:

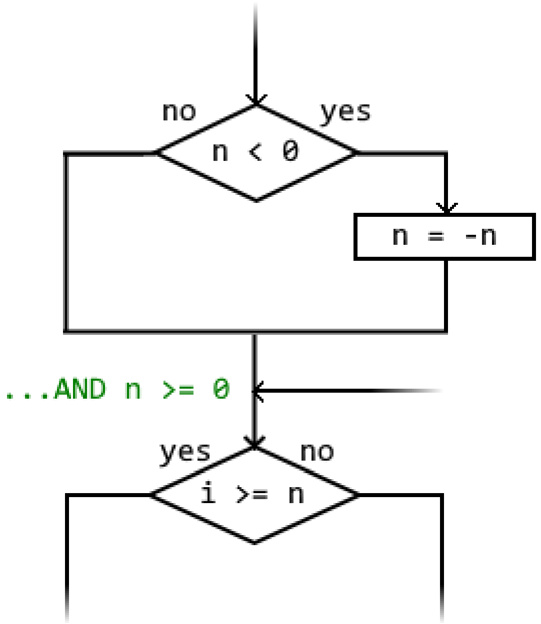
- compared to previous examples, $x = 2i$ would **not** suffice, since we cannot imply that $i \geq n$ (we need to show that, at that moment, $i = n$ holds)
- we make our assertion stronger: $(i < n \wedge x = 2i) \vee (i \geq n \wedge x = 2n) \Longleftarrow (i < n \wedge x = 2i) \vee (i = n \wedge x = 2n)$
- eventually, we reach $x = 2i \wedge i \leq n$, but $x = 2i$ **still** does not imply this assertion, meaning we must include $i \leq n$ in our assertion ($I \equiv x = 2i \wedge i \leq n$)



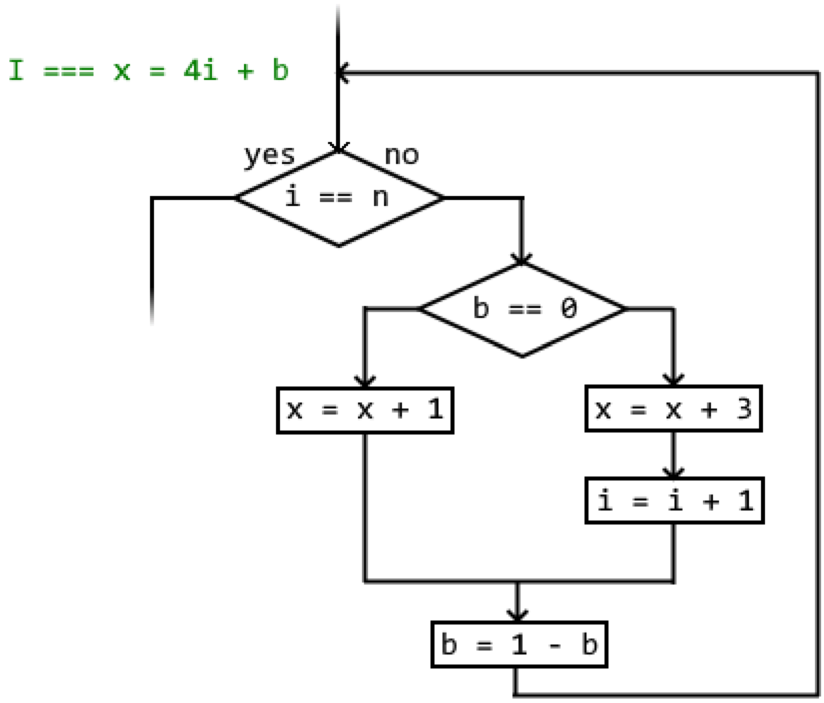
- **Tip 6**: if certain program inputs are **restricted** (e.g. having to take the absolute value of an input n to only allow positive integers), these restrictions about input variables should be included in the loop invariant
  - *example*:
    - this program doesn't work for some negative n...



    - as such, some construct is most likely included before the loop to handle problematic inputs, which **must be accounted for** in the loop invariant



- **Tip 7**: if the variable in our loop invariant **depends on some other value**, it needs to be included in some way in our loop invariant - if the value constantly fluctuates between one value and another, a simple formula will do, otherwise, do case-by-case if-then-else analysis (e.g. $(c1 \implies ...) \land ... \land (c_n \implies ...))$
  - *example*:
    - x is dependent on the value b, which is either 0 or 1 in any iteration (but 0 when we're done)
    - as such, a simple formula $I \equiv x = 4i + b$ suffices



- **Tip 8**: when proving termination, r **must be included** in the loop invariant
- **Tip 8.5**: when proving termination, **all variables required** for calculating r must be included in the loop invariant