

More Notation

Curt Clifton

Rose-Hulman Institute of Technology

Tomorrow

- * Class will meet but is **optional**
- * HW 8 due Monday, 6 a.m.
- * Very challenging for some people
- * Class tomorrow will be a work period on the homework with me available to help

While Rule

IF:

//@ assert P && B;

S;

//@ assert P;

AND

P && !B ==> Q

THEN:

//@ assert P;

while (B) {

//@ assert P && B;

S;

//@ assert P;

}

//@ assert P && !B;

//@ assert Q;

Another Example

* Prove Correctness:

```
        //@ assert b > 0;  
r = 0;  
c = b;  
while (c > 0) {  
    r = r + a;  
    c = c - 1;  
}  
        //@ assert r == a * b;
```

Solution

LOOP INVARIANT: $c \geq 0 \ \&\& \ r == (b - c) * a$

```

    //@ assert b > 0;
10: //@ assert b >= 0 && 0 == (b - b) * a;
    r = 0;
9: //@ assert b >= 0 && r == (b - b) * a;
8: c' = b; // tick-trick
2: //@ assert c' >= 0 && r == (b - c') * a;
    while (c > 0) {
3: //@ assert c >= 0 && r == (b - c) * a && c > 0;
7: //@ assert c > 0 && r + a == (b - c) * a + a;
6: r' = r + a; // tick-trick
5: //@ assert c - 1 >= 0 && r' == (b - c + 1) * a;
4: c' = c - 1; // tick-trick
1: //@ assert c' >= 0 && r == (b - c) * a;
    }
11: //@ assert c >= 0 && r == (b - c) * a && c <= 0;
12: //@ assert c == 0 && r == (b - c) * a;
    //@ assert r == a * b;
```

Recording Loop Invariants in Code

- * **maintaining** clauses
 - * Give loop invariant
 - * Conjoin the expressions to get the complete invariant
- * **decreasing** clause
 - * Gives *bound function* for loop

Example

```
//@ assert b > 0;
r = 0;
c = b;
/*@ maintaining c >= 0;
   @ maintaining r == (b - c) * a;
   @ decreasing c;
   @*/
while (c > 0) {
  r = r + a;
  c = c - 1;
}
//@ assert r == a * b;
```

LOOP INVARIANT

$c \geq 0 \ \&\& \ r == (b - c) * a$

**$c \geq 0;$
 $r == (b - c) * a;$**

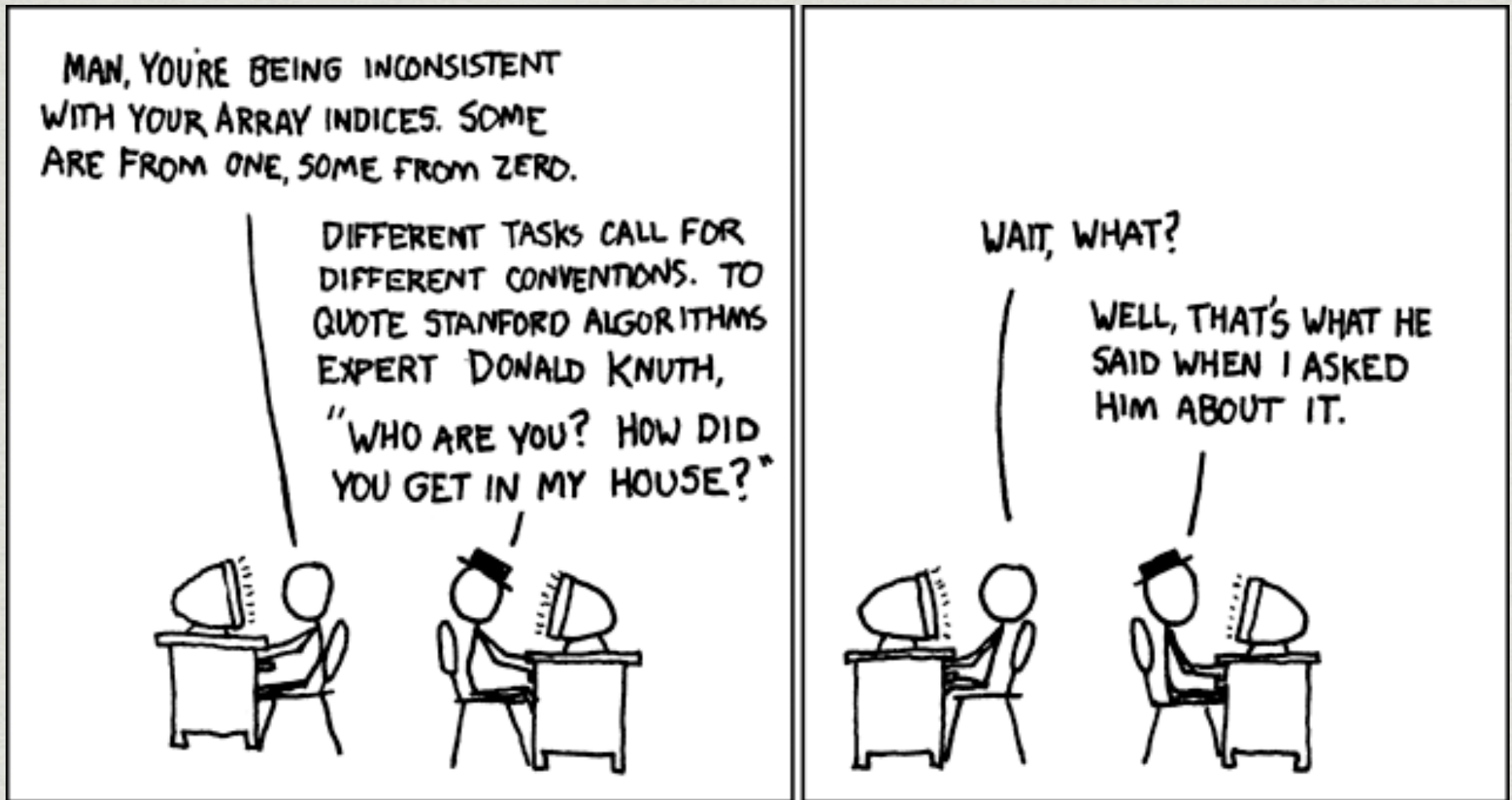
**TERMINATION PROOF:
BOUND FUNCTION C
HAS A LOWER BOUND AND
IS STRICTLY DECREASING**

Termination Example

```
year = 1980;
//@ decreasing days;
while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
```

- * Initially **days** equals elapsed days since Jan. 1, 1980
- * Prove termination:
 - * Bound function?
 - * Lower bound?
 - * Strictly decreasing?

Cartoon of the Day



HIS BOOKS WERE KINDA INTIMIDATING; REPELLING DOWN THROUGH HIS SKYLIGHT SEEMED LIKE THE BEST OPTION.

Quantification in JML

- * $(\forall \text{int } k; 0 < k \ \&\& \ k < a.\text{length}; a[k-1] \leq a[k])$
 - * “The array **a** is sorted.”
 - * Vacuously **true** if range predicate is **false**
- * $(\exists \text{int } j; 0 \leq j \ \&\& \ j < b.\text{length}; \text{isPrime}(b[j]))$
 - * “Some element in the array **b** is prime”
 - * Assuming **isPrime()** does primality testing
 - * Vacuously **false** if range predicate is **false**

More Quantification

- * `(\max int k; lo <= k && k <= hi; a[k])`
- * “The largest element in `a[lo], ..., a[hi]`.”
- * Or the smallest value of the correct type if the range is empty
- * `\min` expressions are similar

Summing Up

- * `(\sum int j; 0 <= j && j < b.length; Math.abs(b[j]))`
- * “The sum of the absolute values of the elements array **b**.”
- * Zero if the range predicate is **false**

Reminder: Tomorrow

- * Class will meet but is **optional**
- * HW 8 due Monday, 6 a.m.
- * Very challenging for some people
- * Class tomorrow will be a work period on the homework with me available to help