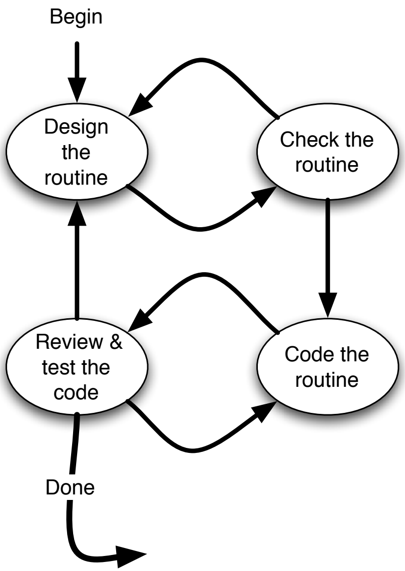
Pseudocode

Dr. Hyrum Carroll

September 15, 2016

Basic Steps in Routine Building



Pseudocode - Program Design Language

- use English statements which describe specific operations
- avoid using programming syntax - keep the description at a high enough level so you aren't thinking in the language
- describe the meaning of the approach rather than the implementation
- start with descriptions at a high level
- provide more details until you can immediately write code from the pseudocode

Pseudocode is an optimal way to design routines.

Good vs Bad Pseudocode

- BAD

```
if (required_node_list(jnode) == 0) then
  required_node_list(jnode) = 1

test the nodes location using if statements
 and subdivide if needed

do i= 1, ndivnodes
  do k = 1, nd(i)
 nd(j++) = d(k)
```

- GOOD

```
mark the node as a "terminal node" if it
 is not already a intermediate

subdivide the node if it is not
 fully in the search region

loop through the divided nodes and
```

Why use Pseudocode

- Pseudocode makes reviews easier
- Pseudocode supports iterative refinement (high-level design $->$ pseudocode $->$ low-level source code)
- Pseudocode makes it easier to change design (catches errors as early as possible)
- Pseudocode can be used as descriptive comments
- Pseudocode is easier to maintain than other forms of design documentation

Pseudocode makes sense.

Pseudocode in Practice

- Use Pseudocode to design your routine
- Write Pseudocode using an editor as comments in your code. Describe the meaning of approach, not implementation
- Check the Pseudocode and make sure it makes sense
- Refine the Pseudocode
- Write the code around the comments
- Check to makes sure the code is correct