Chapter 16: Controlling Loops

Selecting the Kind of Loop

- In most languages, youll find a few kinds of loops
    - The counted loop is performed a specific number of times
    - The continuously evaluated loop that does not know ahead of time how many times it will be executed, and tests whether it is finished on each iteration
    - The endless loop executes forever once it has started
        - Pacemakers, microwave ovens, cruise controls
    - The iterator loop performs its action once for each element in a container class
- The kinds of loops are differentiated first by flexibility – whether executes a specific number of times or tests for completion each iteration
- The kinds of loops are also differentiated by location of the test for completion
    - Beginning, middle or end
- Flexibility and the location of the test determine the kind of loop to choose as a control structure

When to Use a while Loop

- While loops are flexible, because we don't know how many times it will execute
- Loop with test at beginning
- Loop that test at the end
    - Loop that executes at least one time

When to Use a Loop-With-Exit Loop

- A Loop in which the exit condition appears in the middle of the loop rather than at the beginning or the end
- Put all exit conditions in one place
- Use comments for clarification
- This is the preferred kind of loop control

When to Use a for Loop

- Good choice when you need a loop that executes a specified number of times
- Good for simple activities that don't require internal loop controls

Controlling the Loop

- Factors that help avoid problems
    - 1) Minimize the number of factors that affect the loop
    - 2) Treat the inside of the loop as if it were a routine
        - Surrounding program knows the control conditions but not the contents

Entering the Loop

- Enter the loop from one location only

- Put initialization code directly before the loop
- Use while (true) for infinite loops
- Prefer for loops when theyre appropriate
- Don’t use a for loop when a while loop is more appropriate

Processing the Middle of the Loop

- Use clear visuals that you are inside a loop
- Avoid empty loops
- Keep loop housekeeping chores either at beginning or end of the loop
- Make each loop perform only one function

Exiting the Loop

- Assure yourself that the loop ends
- Make loop termination condition obvious
- Avoid code that depends on the loop index’s final value
- Consider using break statements instead of Boolean flags in while loops
- Putting multiple break conditions into separate statements and placing them near the code that produces the break can reduce nesting and make code more readable
- Be wary of a loop with a lot of scattered breaks through it

Checking Endpoints

- A single loop has three cases of interest
    - The first case
    - An arbitrary middle case
    - A last case
- CHECK THESE
    - Willingness to perform these checks is a key difference between efficient and inefficient programmers

Using Loop Variables

- Use ordinal or enumerated types for limits on both arrays and loops
- Use meaningful variable names to make nested loops readable
    - And avoid loop index cross-talk
- Limit the scope of the loop index variables to the loop itself

How Long Should a Loop Be?

- Make your loops short enough to be viewed all at once
    - Typically about 50 lines
- Limit nesting to three levels
- Move loop innards of long loops into routines
- Make long loops especially clear

Creating Loops Easily – From the Inside Out

- General process
    - Start with one case
    - Write pseudocode in comments
    - Code that case with literals
    - Indent it, and put a loop around it
    - Replace the literals with loop indexes or computed expressions
    - Put another loop around that
    - Replace more literals
    - Repeat as many times as needed
    - When finish, add all necessary initializations
- The idea is start at simplest case, then work outward to generalize it
    - Coding from the inside out