CS1110   27 October 2009
while loops
Reading: today: Ch. 7 and ProgramLive sections. For next time: Ch. 8.1-8.3

Prelim 2. Tuesday, 10 November, 7:30PM
If you have a conflict, and if you haven’t been contacted about the conflict, please email Maria Witlox mwitlox@cs.cornell.edu by Friday!!!!

If you have a conflict, and if you haven’t been contacted about the conflict, please email Maria Witlox mwitlox@cs.cornell.edu by Friday!!!!

Watch the lectures on www.videonote.com/cornell

Beyond ranges of integers: the while loop

while (<condition>) {
sequence of declarations and statements
}

<condition>: a boolean expression. 
<repetend>: sequence of statements.

In comparison to for-loops, we get a broader notion of “there’s still stuff to do” (not tied to integer ranges), but we must ensure that “condition” stops holding (since there’s no explicit increment).

Canonical while loops

// Process b..c
for (int k = b; k <= c; k++) {
    Process k;
}

// Process b..c
while (k <= c) {
    Process k;
    k = k+1;
}

Here’s one way to use the while loop

// process a sequence of input not of fixed size
initializations;
while (<still input left>) {
    Process next piece of input;
    make ready for the next piece of input;
}

Understanding assertions about lists

This is an assertion about v
and k. It is true because
chars of v[0..3] are greater
than ‘C’ and chars of v[6..8]
are ‘Z’s.

0   1   2   3   4   5   6   7   8
X  Y  Z  X    A  C  Z  Z  Z

v
≥
C   ?          all Z’s          k
0        3           k                  8
v
≥
W       A C     all Z’s          k
4

Indicate whether each of these 3 assertions is true or false.

The while loop: 4 loopy questions. Allows us to focus on one thing at a time and thus separate our concerns.

// Set c to the number of ‘e’s in String s.
int n= s.length();
k= 0; c= 0;
//  inv: c = #. of ‘e’s in s[0..k-1]
while (k < n) {
    if (s.charAt(k) == ’e’)
        c= c + 1;
    k= k+1;
}
// c = number of ‘e’s in s[0..n-1]

1. How does it start? (how)
does init. make inv true?)
2. When does it stop? (From the invariant and the falsity of loop condition, deduce that result holds.)
3. (How) does it make progress toward termination?
4. How does repetend keep invariant true?

We add the postcondition and also show where the invariant must be true:
initialization;
while ( B ) {
    repetend
}

1. How does loop start? Initialization must truthify invariant P.
2. When does loop stop? At end, P and ‘B’ are true, and these must imply R. Find ‘B’ that satisfies ‘P & & ‘B’ => R.
3. Make progress toward termination? Put something in repetend to ensure this.
4. How to keep invariant true? Put something in repetend to ensure this.
Appendix examples: Develop loop to store in x the sum of 1..100.

We’ll keep this definition of x and k true:

\[ x = \text{sum of } 1..k-1 \]

1. How should the loop start? Make range 1..k-1 empty: \( k=1; \ x=0; \)
2. When can loop stop? What condition lets us know that x has desired result? When \( k == 101 \)
3. How can repetend make progress toward termination? \( k= k+1; \)
4. How do we keep def of x and k true? \( x= x + k; \)

Four loopy questions:

\[
\begin{align*}
&k= 1; \ x= 0; \\
&// \text{ invariant: } x = \text{sum of } 1..(k-1) \\
&\text{while} \ ( k != 101) \ { \\
&\quad x= x + k; \\
&\quad k= k + 1; \\
&}\ // \ { x = \text{sum of } 1..100 } \\
\end{align*}
\]

Building a fair coin from an unfair coin

John von Neumann: building a “fair coin” from an unfair coin

loopy questions:
1. P is true initially
2. When it stops, R is true
3. But we can’t prove that the loop makes progress toward termination!
4. Can’t get something for nothing

Unfair flip produces heads with some probability \( p, 0 < p < 1 \)

Iterative version of logarithmic algorithm to calculate \( b^{*c} \) (we’ve seen a recursive version before):

\[
\begin{align*}
&x = \text{result of flipping a fair coin (heads/tails is true/false) } ^{*} \\
&\text{public static boolean fairFlip() } \{ \\
&\quad \text{boolean } f1= \text{new unfair flip; } \\
&\quad \text{boolean } f2= \text{new unfair flip;} \\
&\quad \text{\textit{invariant: } P: f1, f2 contain results of } 2 \text{ unfair flips, and } \\
&\quad \text{in all previous flips, } f1 \text{ and } f2 \text{ were the same } ^{*} \\
&\quad \text{while} \ ( f1 == f2) \ { \\
&\quad\quad f1= \text{new unfair flip; } \\
&\quad\quad f2= \text{new unfair flip;} \\
&\quad } // P \text{ and } f1 != f2 \\
&\quad \text{return } f1 \ \&\& \ f2 \\
&\} // { x = b^{*c} } \\
\end{align*}
\]

Roach infestation

\[
\begin{align*}
&x = \text{number of weeks it takes roaches to fill the apartment -- see p 244 of text}\} \\
&\text{public static int roaches() } \{ \\
&\quad \text{double} \ \text{roachVol}= .001; \ // \text{Space one roach takes} \\
&\quad \text{double} \ \text{aptVol}= 20*20*8; \ // \text{Apartment volume} \\
&\quad \text{double} \ \text{growthRate}= 1.25; \ // \text{Population growth rate per week} \\
&\quad \text{int} \ w= 0; \ // \text{number of weeks} \\
&\quad \text{int} \ pop= 100; \ // \text{roach population after } w \text{ weeks} \\
&\quad \// \text{ before week } w, \text{volume of roaches < aptVol} \\
&\quad \// \text{ after week } w, \text{volume of roaches > aptVol} \\
&\quad \text{while} \ ( \text{aptVol} > \text{pop} * \text{roachVol}) \ { \\
&\quad\quad \text{pop}= \text{(int) (pop} * \text{growthRate); } \\
&\quad\quad w= w + 1; \\
&\quad // \text{Apartment is filled, for the first time, at week } w. \\
&\quad \text{return } w; \\
&\} \ // { x = b^{*c} } \\
\end{align*}
\]

Calculate quotient and remainder when dividing x by y

\[
\frac{x}{y} = q + \frac{r}{y} \\
21/4= 4 + 3/4
\]

Property: \( x = q \times y + r \) and \( 0 \leq r < y \)

\[
\begin{align*}
&\text{// Set } q \text{ to quotient and } r \text{ to remainder.} \\
&\text{Note: } x >= 0 \text{ and } y > 0 ^{*} \\
&\quad \text{int} \ q=0; \ \text{int} \ r= x; \\
&\quad // \text{invariant: } x = q \times y + r \ \text{and } 0 \leq r \\
&\quad \text{while} \ ( r >= y) \ { \\
&\quad\quad r= r - y; \\
&\quad\quad q= q + 1; \\
&\quad } // \ { x = q \times y + r \ \text{and } 0 \leq r < y } \\
\end{align*}
\]