More on Loops

CS 150 – C++ Programming I
Lecture 7

Writing Loops Correctly

- Many programmers think that writing loops correctly is a matter of luck, trial and error, or magic. That is not the case.
  - It is easy to consistently write correct loops
  - You just have to learn how

- I learned the technique I'm going to teach you back in the 1980s when I first started programming
  - Developed by Berkeley Professor Doug Cooper who wrote the book Oh! Pascal!
  - It has helped me as a programmer more than any other thing that I learned in school

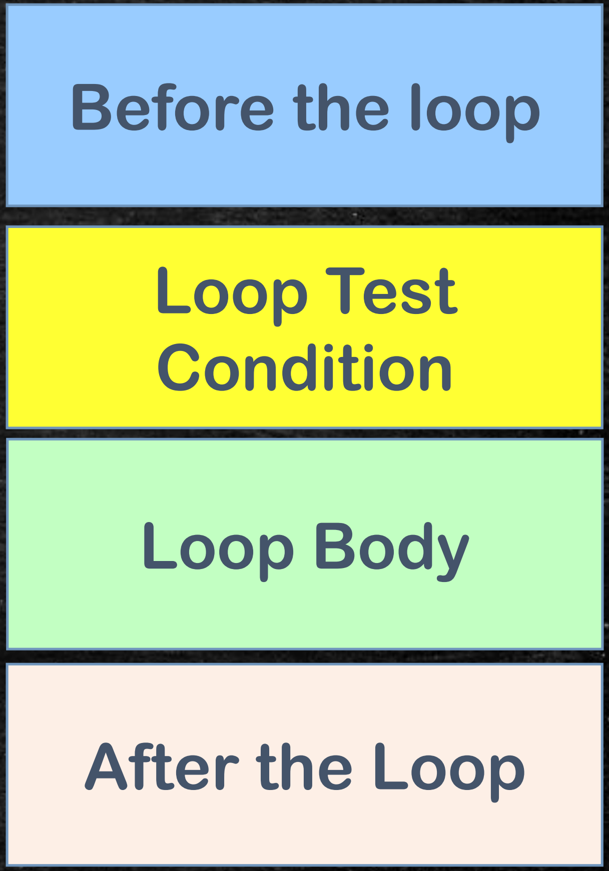
Goal, Bounds & Plan

- Here is a simple problem specification that requires a loop
  - Count the characters in a sentence that ends with a period
- The goal is what the loop is trying to accomplish
  - This loop will produce a count of characters
- The bounds is what makes the loop stop
  - This loop will stop when a period is encountered
- The plan is the set of steps needed to reach the goal

```
– read the first character
  while the character is not the period
      count the character
      read the next character
```

The Loop Topology

- If you look at the loop on the right you'll see four different sections
- Actions that occur before the loop
  - Called the loop's preconditions
- Actions that control the loop (test)
  - This is called the loop's bounds
- Actions that take place in the loop body
  - These are the operations of the loop
- Actions that occur after the loop is over
  - Called the loop's postcondition



Before the loop

Loop Test Condition

Loop Body

After the Loop

Part 1 – The Loop Mechanics

- Think only about what makes the loop work, not the work that the loop does
  - 1. What makes the loop stop (the loop's bounds)

    -
      ```
      while letter is not a period
      ```
  - 2. What setup needed to enter the loop (bounds precondition)

    -
      ```
      str <- the string to check
      letter <- first character in str
      while letter is not a period
      ```
  - 3. What advances the loop towards the bounds

    -
      ```
      while letter is not a period
          letter <- next character in str
      ```

Part 2 – Doing the Work

- The goal of the loop is to produce some information
  - 4. Create variables to hold "answer" (goal precondition)

    -
      ```
      count <- 0 // the result of the loop
      str <- the string to check
      letter <- first character in str
      while letter is not a period
      ```
  - 5. Do work required to update variables (operation)

    -
      ```
      while letter is not a period
          count++ // update the goal
          letter <- next character in str
      ```
  - 6. After, have you reached the goal? (postcondition)

    -
      ```
      count++  // count the period itself
      ```

Necessary and Intentional Bounds

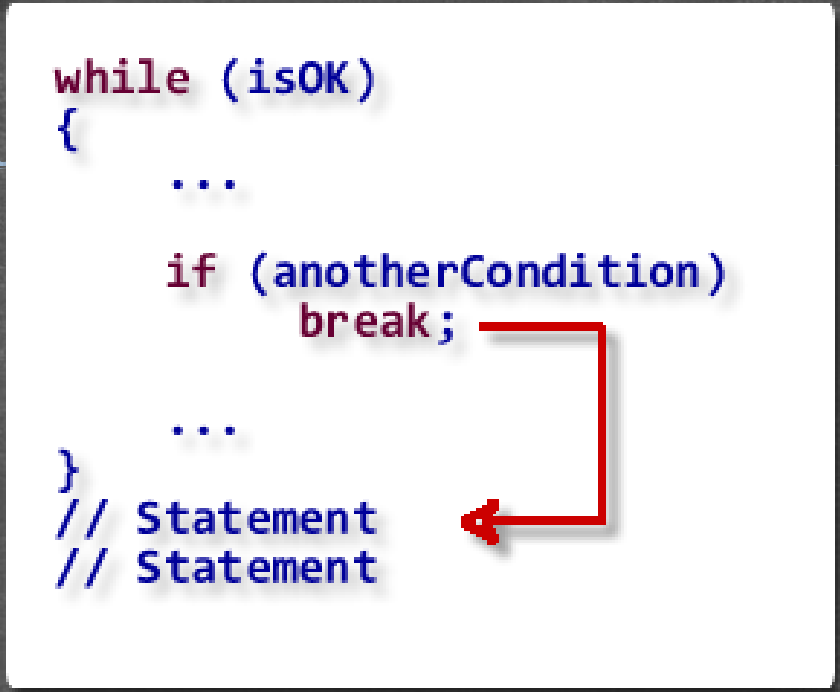
- We have a string "123.25" and we want just "123"
- The loop should stop when you find the "." (sentinel)
  - This is called the intentional bound of the loop

  -
    ```
    size_t i = 0;
    while (str.at(i) != '.')
        ++i;
    ```
- What if the string doesn't contain a period?
  - Then we need an additional (necessary) bounds

  -
    ```
    size_t i = 0;
    while (i < str.size() && str.at(i) != '.')
        ++i;
    ```

The *break* Statement

- *break* jumps out of a *switch* or loop
  - In a loop, *break* jumps to the first statement following the loop body
  - Can make your code clearer when used to construct a loop-and-a-half



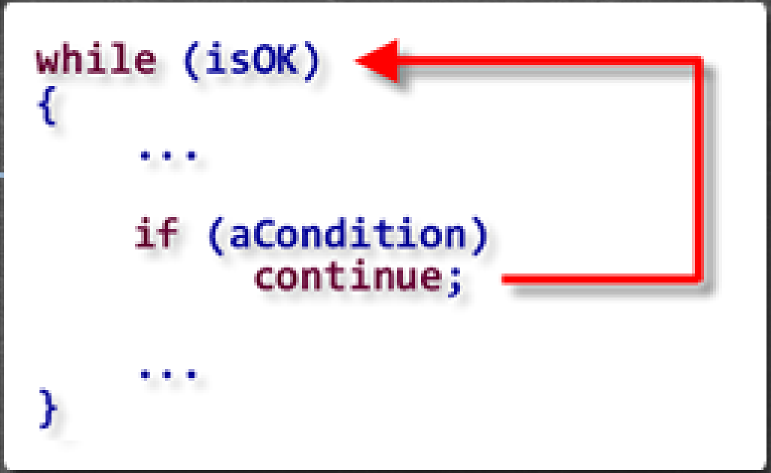
```
while (isOK)
{
     ...

     if (anotherCondition)
          break;

     ...
}
// Statement
// Statement
```

- Available in languages like Ada: *exit when*
  - In C++, use *if* along with *break*

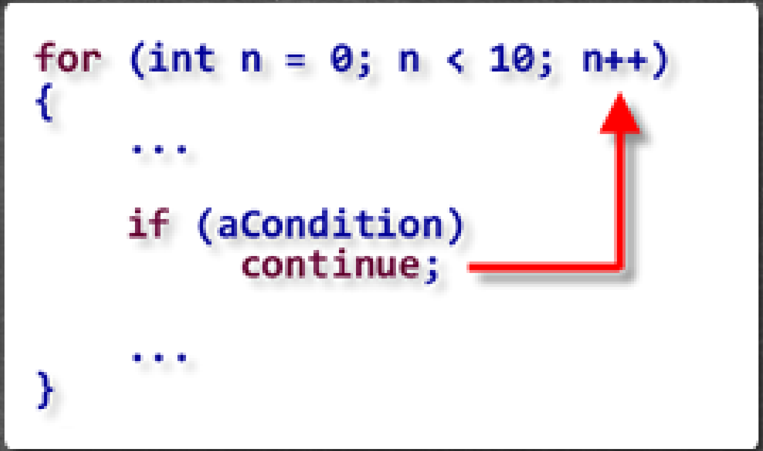
  -
    ```
    size_t i = 0;
    while (i < str.size())  // necessary bounds
       if (str.at(i) == '.') break;
       else ++i;
    ```

The continue Statement

- *continue* only works inside loops
  - Instead of leaving the loop, it starts the next iteration
- In *while* and *do-while* it jumps to the loop test expression
- In *for*, it jumps to the update expression
- Exercise: MoMoney



```
while (isOK)
{
    ...

    if (aCondition)
        continue;

    ...
}
```



```
for (int n = 0; n < 10; n++)
{
    ...

    if (aCondition)
        continue;

    ...
}
```