APS105 Lecture 7
4.1-4.2

Dan Zingaro

February 3, 2010

Feedback from Reading Quiz

Do these really do the same thing?

```
do
  <statement>
while (<expression>);
```

```
while (<expression>)
  <statement>;
```

- "What's the difference between a do and a while? Can we use **only** a while statement?"

Feedback from Reading Quiz...

This loop never executes. Or might it?

```
int y;
while ( y==100)
  printf ( "%d" , y);
```

- "Nothing was hard in those 2 sections, but i really think we should move on faster so we can have enough time to go over the hard chapter . . ."

While-Loops

- C has three looping structures; we start with while-loops

```
while (<expression>)
  <statement>
```

- C first evaluates the `<expression>`
- Then, there are two possibilities, depending on whether `<expression>` is true or false
  - If `<expression>` is false, we jump to the statement following the while-loop
  - If `<expression>` is true, then `<statement>` is executed, and the process repeats by again testing the `<expression>`

Example: While-loops

```
#include <stdio.h>

int main (void) {
  int value;
  printf("Enter an integer (zero to stop): ");
  scanf("%d",&value);
  while (value != 0) {
    printf("%d %d\n",value,value*value);
    printf("Enter an integer (zero to stop): ");
    scanf("%d",&value);
  }
  return 0;
}
```

- Why do we have the `scanf` before the loop?
- How else could we have written `value != 0`?
- Is it possible for this loop to execute zero times?

Conceptest: Equal Consecutive Pairs

Write a program that prompts the user for a sequence of integers, until 0 is entered. The program counts and prints the number of times that consecutive values are equal. For example, if the input is 3 6 7 7 4 4 4 6 0, the program should print 3.

- Why three? What are they?

Conceptest: Equal Consecutive Pairs...

```
(I)   int num, old, consec = 0;

(II)  }
      printf ("%d\n", consec);

(III) while (num > 0) {

(IV)  old = num;

(V)   if (old == num) consec++;

(VI)  printf ("Enter a number: ");
      scanf ("%d", &num);
```

Which of the following orders is correct?

- A. (I), (III), (IV), (V), (VI), (II)
- B. (I), (VI), (III), (IV), (V), (II)
- C. (I), (VI), (III), (IV), (VI), (V), (II)
- D. (I), (VI), (III), (VI), (IV), (V), (II)

Do-loops

```
do
  <statement>
while (<expression>);
```

- Whereas while-loops test `<expression>` before each execution of `<statement>`, do-loops **first** execute `<statement>`, and then test `<expression>`
- So, a do-loop will first execute `<statement>`, and then test `<expression>`
  - If `<expression>` is false, we jump to the statement following the do-loop
  - If `<expression>` is true, then the process repeats by again executing `<statement>` and testing the `<expression>`

ConcepTest

What is the value of `x` after this code executes?

```
int x = 4;
do {
  x -= 5;
  x++;
} while (x >= 0);
```

- A. -1
- B. -2
- C. -3
- D. -4
- E. 0

While vs. Do

- It's often a matter of preference which looping structure we use
- Sometimes, one loop will be more clear than the others
- e.g. let's rewrite the squaring program using a do-loop