**Using *while* and *for* Loops, Other Loop Examples**

David LaCivita

CTIM 281 Software Design and Development

Professor Chuck Bisson

October 13, 2020

**Using while and for Loops**

Using loops in programming can greatly improve a programmer’s efficiency. Knowing when to use a **for** loop or a **while** loop can improve efficiency even more. A **while** loop executes a set of instructions while a condition remains true. It is best to use a **while** loop when you don’t know the number of iterations the loop will cycle through. This may be because the number may depend on the user’s input or an unknown number of entries in a file. A **for** loop is best used when you know exactly how many times a loop will execute. Every loop must contain a control variable and a **for** loop must initialize, test, and alter the control variable in a single statement.

A simple situation for using a **for** loop would be a program used to print all of the odd numbers between 1 and 100.

start

Declarations

num value

for value = 1 to 100 step 2

output value

endfor

stop

**Other Loop Examples**

Loops can be used anywhere and they can be combined and nested. To simplify instructions for sorting a jar of coins you could use something like this:

// You have a jar with a lot of coins.

// The coins are either dimes, pennies, nickels, or quarters.

// The coins are sized 1,2,3, or 4 respectively.

// The coin counter has 4 slots, one to fit each size of coin.

start

num coinJar

num coinSize

num slotSize

while coinJar <> 0

for coinSize 1 to 4 step 1

for slotSize = 1 to 4 step 1

fit coin into slot

endfor

endfor

endwhile

stop

A slightly less technical example might be something like:

while there is dog poop in the yard

scoop poop

endwhile

Loops are essential to efficient programming and can be useful in everyday life. When a programmer knows how to effectively use **while** and **for** loops the code can be written faster and compiled quicker saving time and money.