**Algorithms**

What is an algorithm? An algorithm is a set of ordered steps for solving a problem.

People create or write algorithms all the time. For example, which of the following algorithms have you already created?

* The car is dead. How do I get to work/school today?
* I need to pick up on the way home today my laundry, some milk, and a book I ordered at the bookstore. How can I do this with the least amount of travel/trouble?
* I want to order new wall-to-wall carpeting for my living room. How do I calculate square yards?
* I’m traveling to Europe next month. The average temperature will be 19 degrees Celsius. What is that in Fahrenheit (what should I pack)?
* How do I write a program to check all the answers on standardized tests to look for cheating patterns?
* What is the calculation necessary to send a rocket into a particular orbit?
* How do I use a computer program to model human genetics in order to solve large difficult programs?

As you can see, one can create an algorithm for practically anything.

For an algorithm to function properly, the following statements must be true:

* The steps must be precise and individual
* The steps must be ordered
* The steps must not be ambiguous
* The steps must reach a solution in a finite amount of time

As we discussed in class, algorithms consist of four basic types of operations or instructions:

1. Sequential operations, e.g. Add 1 cup of butter; Subtract the amount of the check from the current account balance; Set the value of x to 1
2. Conditional operations, e.g. If the mixture is too dry, then add ½ cup water; If the current account balance < 0, then transfer funds immediately
3. Iterative operations, e.g. Repeat the previous two steps until the mixture has thickened; Repeat the following five steps until there are no more checks to be processed; Repeat steps 1,2 and 3 until the value of y is equal to +1
4. Calling a pre-existing algorithm (or function), e.g. Make a *reduction* with your sauce; date := getDate(); =normdist(x, mean, stdev, true)

Let’s look at each of these types of instructions showing examples of each. Here is a simple algorithm (written in English) that calculates the mileage, or Average Miles Per Gallon, for your car. Note that the algorithm contains only sequential instructions:

1. Get values for *gallons used*, *starting mileage*, *ending mileage*
2. Set the value of *distance driven* to *ending mileage* – *starting mileage*
3. Set the value of *average miles per gallon* to *distance driven /* *gallons used*
4. Print the value of *average miles per gallon*
5. Stop

Here is an algorithm for calculating a solution to the quadratic formula:

 x = (-b ± sqrt(b2 - 4ac)) / 2a

1. Calculate b2
2. Calculate 4 \* a \* c
3. Subtract 4ac from b2
4. Take the square root of b2 - 4ac
5. Add square root value to -b
6. Subtract square root value from -b
7. Calculate 2 \* a
8. Divide -b + square root value by 2a
9. Divide -b - square root value by 2a
10. Display the result
11. Stop

Most of the time when performing calculations we need to store temporary values into what we call variables. The naming conventions for variables differ from programming language to programming language. Since we aren’t using a real programming language yet, we’ll just make up some variable names such as Var1, Var2, Var3, etc. We’ll see examples of this shortly.

Now let’s take a look at the conditional statement. The most popular form of conditional is the *if else* statement. The basic form looks like this:

 if some condition is true then

 do this statement

 and this statement

 else

 do this statement

 and this statement

 end if

For example, let’s say you are online and if you order more than $100 in books, the shipping is free. The algorithm to handle this might look like the following:

 if OrderTotal > 100.00 then

 set ShippingCost to 0

 print “Thank you for your order over $100.00!”

 else

 set ShippingCost to 5.45

 print “We charged you $5.45 for shipping.”

 end if

Even though the above algorithm is still not in a programming language, it may not be very English-like. Here is what we are saying in English:

 if the total of you order is greater than $100.00 then

 the shipping cost is zero and

 tell the customer “Thank you for your order over $100.00”

otherwise (if the order is not greater than $100.00, or stated another way, if the order is less than or equal to $100.00) then

 the shipping cost is $5.45 and

 tell the customer “We are charging you $5.45 for shipping.”

 end of the if statement

We can modify the earlier example of calculating the mileage of your car by adding an *if else* statement:

1. Get values for *gallons used*, *starting mileage*, *ending mileage*
2. Set the value of *distance driven* to *ending mileage* – *starting mileage*
3. Set the value of *average miles per gallon* to *distance driven /* *gallons used*
4. Print the value of *average miles per gallon*
5. If *average miles per gallon* is greater than 25.0 then

Print the message “You are getting good gas mileage!”

Else

Print the message “You are NOT getting good gas mileage. Buy a Chevy Volt”

 End if

1. Stop

The third type of statement is the loop. A loop is inserted into an algorithm when you want to repeat one or more statements. For example, let’s consider the algorithm that might be used to calculate the payroll for all the employees at XYZ Corporation:

1. Get the first employee’s data
2. Calculate first employee’s gross pay
3. Calculate first employee’s state income tax
4. Calculate first employee’s federal income tax
5. Calculate first employee’s take home pay
6. Print first employee’s pay check
7. Get the second employee’s data
8. Calculate second employee’s gross pay
9. Calculate second employee’s state income tax
10. Calculate second employee’s federal income tax
11. Calculate second employee’s take home pay
12. Print second employee’s pay check
13. and so on for ALL the employees in the company!

But if there are 100s or 1000s of employees at this company, someone is going to have to write a LOT of repetitious statements to complete this algorithm. Instead, let’s use a loop. A very common loop is the while loop. It has the following form:

 while some condition is true, perform the following steps

 step 1

 step 2

 step 3

 :

 end loop

The steps inside the loop are repeated over and over as long as the condition is true. To solve our employee payroll problem with a while loop:

While there are employees to process

Get the employee’s data

Calculate employee’s gross pay

Calculate employee’s state income tax

Calculate employee’s federal income tax

Calculate employee’s take home pay

Print employee’s pay check

end loop

Notice how this while loop can repeat as many times as there are employees to process.

Now we will create an algorithm which uses all three types of statements: sequential, if else, and the loop. Write the algorithm to examine student scores on an exam. You want to count how many students passed (>=70) and how many students failed (<70). When you hit the end of the list, print the total number of students that passed and the total number of students that failed. We’ll need two counters: one for counting how many students passed, and one for counting how many students failed.

 Set StudentsPassed counter to 0

 Set StudentsFailed counter to 0

 While there are student scores to process

 If the student score is >= 70.0 then

 Add 1 to StudentsPassed

 else

 Add 1 to StudentsFailed

 end if

 end loop

 Print StudentsPassed counter

 Print StudentsFailed counter

Up to this point all the algorithms that we have created have not been based on a real programming language. Let’s take a look at a couple examples of algorithms using a real programming language. We could use something like Java or Python, but even those are fairly elaborate. Instead, let’s use a very English-like language: Visual Basic.

Here is an example of an *If statement*:

if Row < 100 Then

Cells(Row, 2) = 1

Else
Cells(Row, 2) = 0

end If

How about an example of a *while* loop?

Counter = 0

while Counter < 20

 Counter = Counter + 1

end while

Another type of loop other than a *while* loop is a *for* loop. The *for* loop is what we call a finite loop, while the *while* loop is an indefinite loop. In other words, we tell the *for* loop exactly how many times it is supposed to loop. The *while* loop will loop until come condition is no longer true. For example, if we want a piece of code to iterate exactly 50 times, we could write the following:

for Row = 1 To 50

 Y = rnd(a)

X = X + Y

next Row