**COLLECTION OF PROBLEM-SOLVING ACTIVITIES**

**CS: 200**

**BYRON LAFERRIERE**

**UNIVERSITY OF SOUTHERN NEW HAMPSHIRE**

**PSEUDOCODE FOR PAINTING A WALL**

#Prompt user to enter height and width

input height  
input width  
#Multiply height by width to find total area in square feet

area = height \* width  
# Display Wall area in square feet

output ‘ Wall area: ‘ + area + ‘ square feet. ‘  
# Prompt user to enter known paintpersqft 350sq = 1 gallon

input paintPerSqFt  
# Multiply paintpersqft by wall area to find how many gallons will be needed(paintNeeded)

paintNeeded = paintPerSqFt \* area  
#Display paintNeeded

output ‘ Paint needed: ‘ + paintNeeded + ‘ gallons. ‘  
# Display number of cans needed for gallons of paint

cans = paintNeeded + 1  
output ‘Cans needed: ‘ + cans + ‘ can(s). ‘  
#Prompt user to enter a desired color to find total cost of paint

# Red = 35 Blue = 25 Green = 23

input color  
output ‘ Cost of purchasing ‘ + color + ‘ paint: $ ‘ + price(per can) \* cans

END

This was a simpler pseudocode code to complete, so it’s important to remember to follow the best practice “Readability Counts.” It can be easy to overcomplicate your logic on easier tasks and get wordy. Keeping this short code simple and direct is the best practice for a task of this manner. Learning to keep true to this practice will benefit me down the road as I progress through my career. There was no need to use a loop in this code since the area of the wall would be predefined by the user. I was able to simply define variables and allow the program to determine how many gallons of paint would be needed to cover the user’s wall area. Using the example of 350sq ft = 1 gallon of paint, the program would compare the users inputted wall area versus the known coverage of one gallon of paint. I did feel however, that the program was a bit redundant in requesting the extension of converting gallons to cans, when paint is sold by the gallon in cans usually.

In the end, since the variables had been set and pre-determined, the program was a rather simple input/output example. This was an excellent demonstration to begin with to avoid overwhelming students who are taking on coding for the first time. This exercise provided an excellent introductory example into the world of coding.

**Pseudocode for Auto Shop**

Create dictionary for value of car wash options

Assign option 1 and 2 to open string

Assign Total to Zero to keep integer value

Display options available and price by Davy’s auto shop

Assign value to variables in dictionary (#including ‘- == 0’)

Prompt user input for option 1 and 2 from menu

WHILE both input is not ‘-‘ : (# While loop will allow program to check multiple times for input)

IF two options inputted: (#Use the nested IF loop to check for completed program)

Display service 1 and service 2 that were selected

BREAK (# Use break to end loop and terminate program)

ELSE:

Retrieve cost of selected service 1 and service 2 including ‘-‘ ( # Even if only one service selected add total cost )

ENDWHILE (# End while loop)

Display cost of service 1 and service 2

END

While writing this control structure for the activity from Module Three, I continuously found myself having to go back and paraphrase what I was trying to say. Now that I am beginning to understand control structures better, it is getting easier to put my thoughts to paper. I had to keep the best practice in mind, “Simple is better than complex.” By going back and trying to paraphrase my thoughts I found myself writing more clearly and directly. “Remembering to write for people and not computers” is another best practice that I found which I will begin keeping in mind while tackling tasks. This means keeping consistency throughout the code, as well using original names, that are concise, to ensure it easy to follow along for other readers.

When it came to the decision of how I would write out the control structure, there were many possible routes that could have been taken to create the annotations that were used. I decided to begin by opening with a while loop to check for the null value of ‘-‘ in the input. While both options did not have zero input, the program would run, gather the requested services and display the total cost for the user. From this point we can initiate the IF loop to check for two selected services, to break the program and finalize the total. Otherwise the else will check for an option 1 selection, or an option 2 selection and terminate the program as well. Using the control structures in this manner will allow the program to determine the necessary information, without running an infinite loop searching for the selected options.

**PSEUDOCODE FOR DRAWING A HALF ARROW**

#Display prompt to input arrow base width, arrow base height, and arrow head width

Input arrow base height

Input arrow base width

Input arrow head width

#Use a WHILE loop to only allow program to work if arrowhead width=< arrow base width

WHILE arrowheadwidth <= arrowbasewidth :

#Prompt user to input arrow width under these parameters and assign to variable

Arrowheadwidth = input ( Enter arrow head width)

# Begin a FOR loop to print an asterisk for input assigned to arrow base height

#Use range() to generate integers

FOR i in range(arrowbaseheight)

# Use a nested FOR loop to check arrowbasewidth against arrowbaseheight to find final dimensions of base

#Use range() to generate integers

FOR j in range(arrowbasewidth)

Display arrow base width by arrow base height

END NESTED FOR LOOPS

# Begin a new FOR loop to find the starting width of arrow head

#Use range() to generate integers

FOR i in range(arrowheadwidth)

#Begin a second NESTED FOR loop to decrease original integer width by 1 until arrow is 1 wide. Printing an asterisk per input assigned to arrow head width

#Use range() to generate integers

FOR j in range(arrowheadwidth – i)

Display arrow head width decreasing by one each row until only one asterisk in line

END NESTED FOR LOOPS

END WHILE

I particularly enjoyed this task and completing this pseudocode before typing the code out. Learning to extract what the program parameters will be expected before writing out the code taught me to see the task differently. Instead of guessing and filling in the code needed or missing as the program fails, it helps to put those parameters in first if you know what they are before-hand. For this pseudocode I used the best practice “Explicit is better than implicit.” This program needed very specific instructions to run correctly, otherwise the arrowhead would not display correctly decreasing in size. To ensure that this happened, I used nested for loops to have the program iterate as many times as necessary until only one asterisk displayed at the tip of the arrowhead. A nested for loop was also needed to determine the correct size of the base of the arrow. I used the nested for loop to retrieve the arrow base height by arrow base width. Once the dimensions had been identified in the loop, the program would display the dimensions with corresponding asterisks.

By using the while loop to moderate the parameters of the program, I was able to monitor each iteration of the program under the specifications of arrowheadwidth being smaller than arrowbasewidth before the program will break. This guaranteed that the result produced by running the code would look like an arrow and nothing else. This activity was a very good example of using nested loops and how to use a control structure, like while, to guide a program through iterations.

**References:**

Best Practices For Scientific Computing

Greg Wilson-D. Aruliah-C. Brown-Neil Hong-Matt Davis-Richard Guy-Steven Haddock-Kathryn Huff-Ian Mitchell-Mark Plumbley-Ben Waugh-Ethan White-Paul Wilson - <https://journals.plos.org/plosbiology/article?id=10.1371%2Fjournal.pbio.1001745>

Boston Dynamics

<https://www.bostondynamics.com/>

CS:200 – Computer Science’s Role in Industry

<https://learn.zybooks.com/zybook/SNHUCS200AY16-17>

Pep 20 -- The Zen Of Python

<https://www.python.org/dev/peps/pep-0020/>

Python 3 – Quick Reference Card

<https://www.cs.put.poznan.pl/csobaniec/software/python/py-qrc.html>