**Stepwise Refinement Approach**

The stepwise refinement approach is a methodical approach used to address the difficulty encountered while addressing design algorithms of non-trivial processes (*CodeIT Project*, n.d.). It is an effective consideration when implementing really difficult concepts where the described process is very close to the intended process. The stepwise refinement design is also known as the top-down design because it details precisely the process that needs to be performed with all possible identified circumstances. It is generally documented in an iterative/incremental mode such that new processes are added to refine processes into simplified layers that better present processes at the lowest possible representations.

The stepwise refinement approach can be used to simplify and solve several difficult operations to include cylindrical and spherical functions which are classes of transcendental functions (Dvortsov, 2022).

For the stepwise refinement approach scenarios provided, I will be addressing the check writer that prints in words, amounts given in dollars. To achieve this, I will state the initial solution as understood by the requirements provided by the owner, then I will refine this initial solution. Finally, I will detail to the minutest process the procedural abstraction as presented in the refinement stage. Below is the solution provided:

***Initial solution***

1. Accept amount on check
2. Verify check is not empty
3. Compute length of the amount on check
4. Create array of numbers in words format
5. Verify length is equal to one
6. Iterate through the value

***Refinement***

1. Accept amount on check as string
2. Compute length of the amount on check
3. Verify check amount is not empty

* If length is 0 then, empty string

1. Create arrays of numbers in word format
2. Array from zero to nine
3. Array from ten to nineteen
4. Array of multiples of ten (from twenty and ninety)
5. Array powers of ten (hundred and thousand)
6. If length equals one
7. Print value from the single-digit array
8. Iterate through the value
9. If length greater than or equal to three
10. If check[x]-‘0’ is != to zero
11. Print unitary\_digits[num[x]-‘0’]
12. Print tens\_power[len-3]
13. Decrement the length
14. ElseIf num[x]-‘0’ is equal to one
15. Assign sum is equal to num[x]-‘0’ + num[x]-‘0’
16. Print teo\_digits[sum]
17. Else if num[x]-‘0’ is equal to 2 and num[x+1]-‘0’ Is equal to 0
18. Print twenty
19. Else
20. Assign I = num[x]-‘0’
21. If I greater than zero
22. Print tens\_multiple[i]
23. Increment x by one
24. If num[x]-‘0’ not equal to zero
25. Print single\_digits[num[x]-‘0’]
26. Increment x by 1

***Procedural abstraction***

def WordsFunction(check):

    digit\_length = len(check)

    if digit\_length == 0:

        print("empty string")

        return

    if digit\_length > 4:

        print("Length more than 4 is not supported")

        return

    unitary\_digits = ["zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"]

    tens\_digits = ["", "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"]

    tens\_multiple = ["", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"]

    tens\_power = ["hundred", "thousand"]

    print('Value on check in words is:', end=" ")

    if (digit\_length == 1):

        print(unitary\_digits[ord(check[0]) - 48])

        return

    x = 0

    while (x < len(check)):

        if (digit\_length >= 3):

            if (ord(check[x]) - 48 != 0):

                print(unitary\_digits[ord(check[x]) - 48], end=" ")

                print(tens\_power[digit\_length - 3], end=" ")

            digit\_length -= 1

        else:

            if (ord(check[x]) - 48 == 1):

                sum = (ord(check[x]) - 48 +

                    ord(check[x+1]) - 48)

                print(tens\_digits[sum])

                return

            elif (ord(check[x]) - 48 == 2 and

                ord(check[x + 1]) - 48 == 0):

                print("twenty")

                return

            else:

                i = ord(check[x]) - 48

                if(i > 0):

                    print(tens\_multiple[i], end=" ")

                else:

                    print("", end="")

                x += 1

                if(ord(check[x]) - 48 != 0):

                    print(unitary\_digits[ord(check[x]) - 48])

        x += 1

(Kangralkar, 2022)

**References:**

*Algorithmic Design: Stepwise refinement of algorithms—CodeIT Project*. (n.d.). Retrieved February 28, 2022, from https://www.codeit-project.eu/algorithmic-design-stepwise-refinement-of-algorithms/

Dvortsov, V. N. (2022). *TRANSCENDENTAL EQUATION*. https://www.thermopedia.com/content/1202/

Kangralkar, N. (2022). *Program to convert a given number to words—GeeksforGeeks*. https://www.geeksforgeeks.org/convert-number-to-words/