1b - Intro Algorithms

Noud van Giersbergen

University of Amsterdam

2021-03-31

Objective Intro Algorithms

- Problem Introducution
- Flowcharts
- Pseudocode
- Algorithm

Problem Introduction

- Many quantitative problems are solvable by computers
- Things to consider
    - Analyse
    - Number and types of input
    - Output and its type
    - Constraints
    - Flow chart
    - Pseudocode
    - Algorithm

Problem Statement - Modified Armstrong Numer

A number is considere to be modified Armstrong if it is equal to the sum of its digits raised to the power of the total number of digits in num and it doesn't end with a 0.

For example: 153

$$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$$

Your task is to check if a given number is an Armstrong number or not.

Analysis

- A number is given
- Need to check if given number is armstrong number
- Questions
    - Ask for range of the number
    - Ask for the type of output expected
    - Ask if the total number of digits is given or not

I/O

- Input
  - Type: Integer
  - Number of inputs: 1
- Output
  - Type: Logical
  - Number of outputs: 1

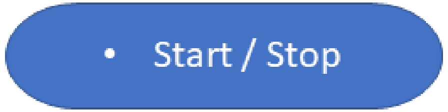
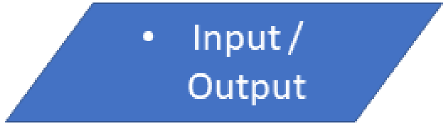
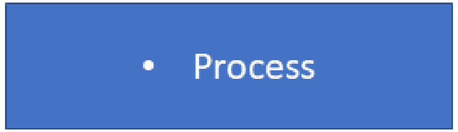
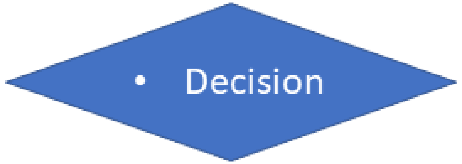
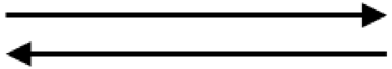
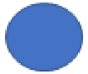
Constraints

- Number must not end with 0
- Conditions of Armstrong number
- Numer of digits is not given

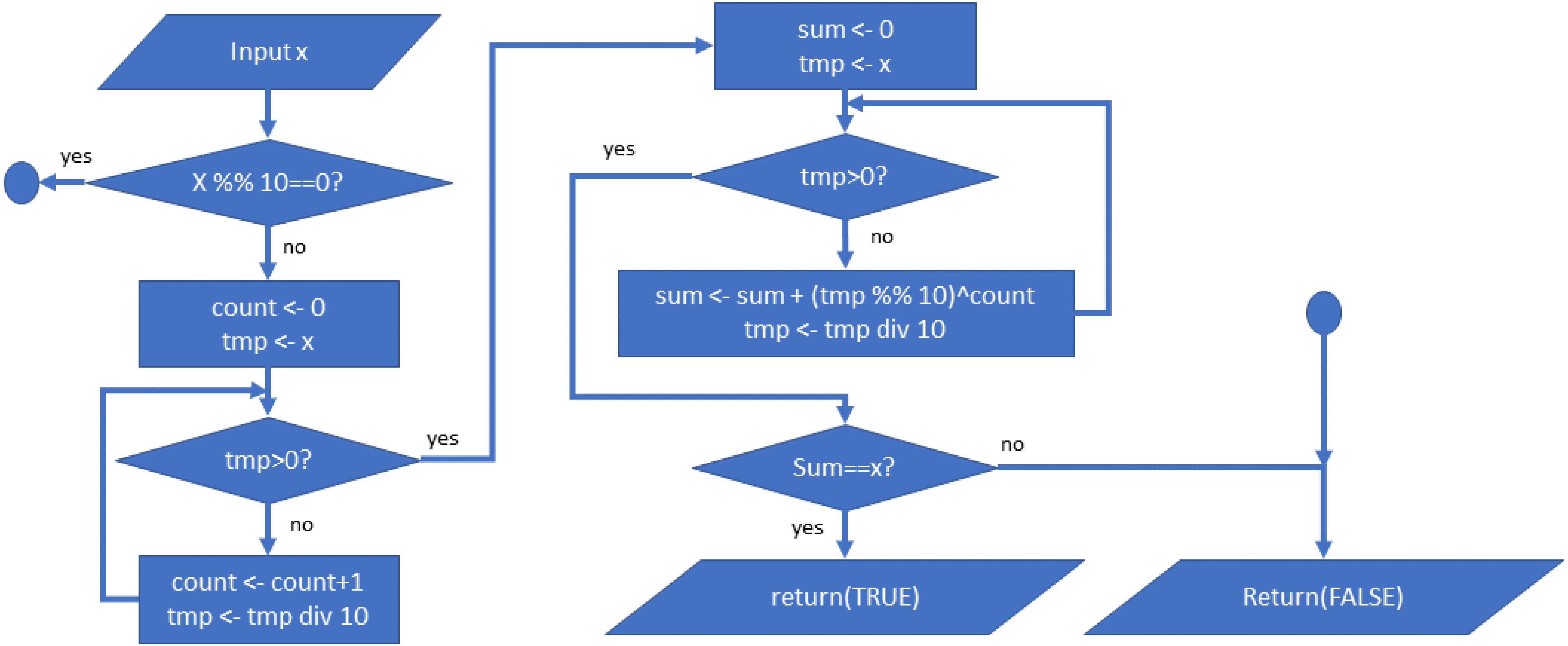
Flow Chart

- Pictorial representation of steps to be performed
- Enable visualization of the problem
- Shows clear data flow with the help of arrows
- Can be used for a non-technical audience too
- Have a definite start and end point

Flow Chart - Symbols

Start and end of a flowchart	 • Start / Stop	Ellipse
Read data (input) or print data (output)	 • Input / Output	parallelogram
Shows a process (e.g. assignments)	 • Process	rectangle
Decision step (if …)	 • Decision	diamond
Connect steps to show the sequence		arrows
On-page Connector		circle

Flow Chart - Modified Armstrong Number



Pseudocode - Modified Armstrong Number

- Input a number
- Check the last digit of the number. If it is 0 then return `FALSE`
- Count the number of digits in the number
- Take each digit and add digit raised to the power of the total number of digits. Add it to the sum variable
- Check if the number is equals to the sum obtained. If so, then return `TRUE` otherwise return `FALSE`

Algorithm

- Def: "an algorithm is a finite sequence of elementary steps leading to the resolution of a problem"
  1. it can be used on different inputs generating corresponding outputs
  2. each step allows a single unambiguous interpretation and is executable in a finite amount of time
  3. whatever the input, its execution eventually stops
- It's more close to actual programming
- It follows programming constructs to some extent
- Algorithm is independent of any programming language
- It is used to analyze time / space complexity

Algorithm - Modified Armstrong Number

```
Input x
if x%%10==0 return false
tmp=x
count=0
while tmp>0
    count++
    tmp=tmp div 10
tmp=x
sum=0
while tmp>0
    sum=sum+(tmp%%10)^count
    tmp=tmp div 10
if sum==x return TRUE
else return FALSE
```

R implementation

```r
Armstrong <- function(x){
  if(x%%10==0) return(FALSE)
  else {
    tmp <- x
    count <- 0
    while(tmp>0){
      count <- count+1
      tmp <- tmp%/%10
    }
    tmp <- x
    som <- 0
    while(tmp>0){
      som <-  som + (tmp%%10)^count
      tmp <- tmp%/%10
    }
    if(som==x) return(TRUE) else return(FALSE)
  }
}
Armstrong(153)
```

```
[1] TRUE
```

R implementation - More efficient

```
Armstrong <- function(x){
  if(x%%10==0) return(FALSE)
  else {
    xstr <- toString(x)
    count <- nchar(xstr)
    som <- 0
    for(i in 1:count){
      som <-  som + as.numeric(substr(xstr,i,i))^count
    }
    if(som==x) return(TRUE) else return(FALSE)
  }
}
Armstrong(153)
```

```
[1] TRUE
```