Problem Solving and Programming
ගැටළු විසඳීම හා පරිගණක ක්‍රමලේඛනය

Anton Gajasinghe
BSc. – University of Kelaniya
PGDIP for IT – University of Peradeniya

Problems ?
ගැටළු ?

- දත්ත ඇති නමුත් තීරණ ගැනීම සඳහා අපට වැදගත් වන්නේ තොරතුරුය.
- තිබෙන දත්ත වලින් අවශ්‍ය තොරතුරු සකස්කර ගැනීම ගැටළුවකි. ගැටළු ඕනෑම තැනක තිබිය හැකිය. මිනිසුන් ගැටළු විසඳන ආකාරය ඉතා සංකීර්ණය.
- නමුත් පරිගණක ක්‍රමලේඛකයෙක් ගැටළු විසඳීමේ දී එම ගැටළුව විසඳන ආකාරය ඉතා සරල පියවර වලට වෙන්කර ගත යුතුය. ඉන්පසු එම පියවරයන් පරිගණක කේත බවට පත් කල හැකිය.

Computer Programming?

Data → Required Information

Need a software to accomplish the task. Programmers develop software using a Computer programming Language

Three Steps to Programming

Good programmers write programs in three steps.

1. **Identify the Problem**
2. **Simplify the Solution**
3. **Code the Solution**

Identify the Problem

We assume, we have to divide students into two houses of the school for inter house meet.

What are the Inputs, Process and Outputs?

We can divide student's admission number by two.

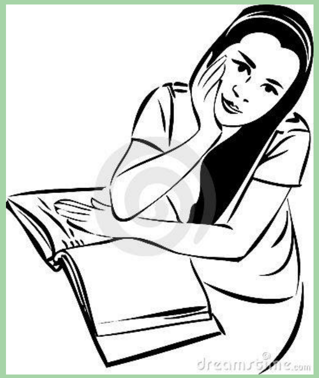
Inputs - Students' Admission Number

Process - Student's Admission Number / 2
If remainder = 0 - House A
If remainder = 1 - House B

Outputs – Student's House Name

How to make a cup of tea?

We need identify,
1. Inputs
2. Process and
3. Outputs
clearly.



Algorithms

යම් කාර්යයක් ඉටුකිරීම සඳහා/යම් ගැටළුවක් විසඳීම සඳහා අනුපිළිවෙලින් ලබා දෙන නිශ්චිත පියවර ගණනකින් සමන්විත උපදෙස් මාලාවක් (set of instructions) ඇල්ගොරිතමයක් ලෙස හැඳින්විය හැකිය.

- ගැටලුවක් විසඳීම සඳහා අනුගමනය කරන ක්‍රමවේදයක් ඇල්ගොරිතමයක් (Algorithm) ලෙස හැඳින්වෙයි.
- ඇල්ගොරිතමයක් රූපමය ආකාරයෙන් (Graphical) හෝ ලිඛිත ආකාරයෙන් (Textual) දැක්විය හැකි ය.
  - රූපමය ආකාරයෙන් - ගැලීම් සටහන්
  - ලිඛිත ආකාරයෙන් - Pseudo Code

ඇල්ගොරිදමයක ලක්ෂණ

- නිවැරදි වීම
- පැහැදිලි වීම
- සීමිත පියවර ගණනක් තිබීම (ආරම්භය හා අවසානය)
- අනුපිළිවෙලක් තිබීම
- එක ගැටළුවකට විසඳුම් කිහිපයක් (ඇල්ගොරිදම) තිබිය හැකිය

ඇල්ගොරිදම

```
Start

Input Admission Number
Admission Number/2

If remainder = 0 then
        House = House A
If remainder = 1 then
        House =House B
end
```

- සංඛ්‍යා දෙකක් එක්තු කිරීම නිවැරදිව සිදුකිරීම සඳහා ඇල්ගොරිදමයක් ලියන්න.
- කහ ඉර මතින් පාර හරහා නිවැරදිව මාරුවන අකාරය දැක්වීම සඳහා ඇල්ගොරිදමයක් ලියන්න.

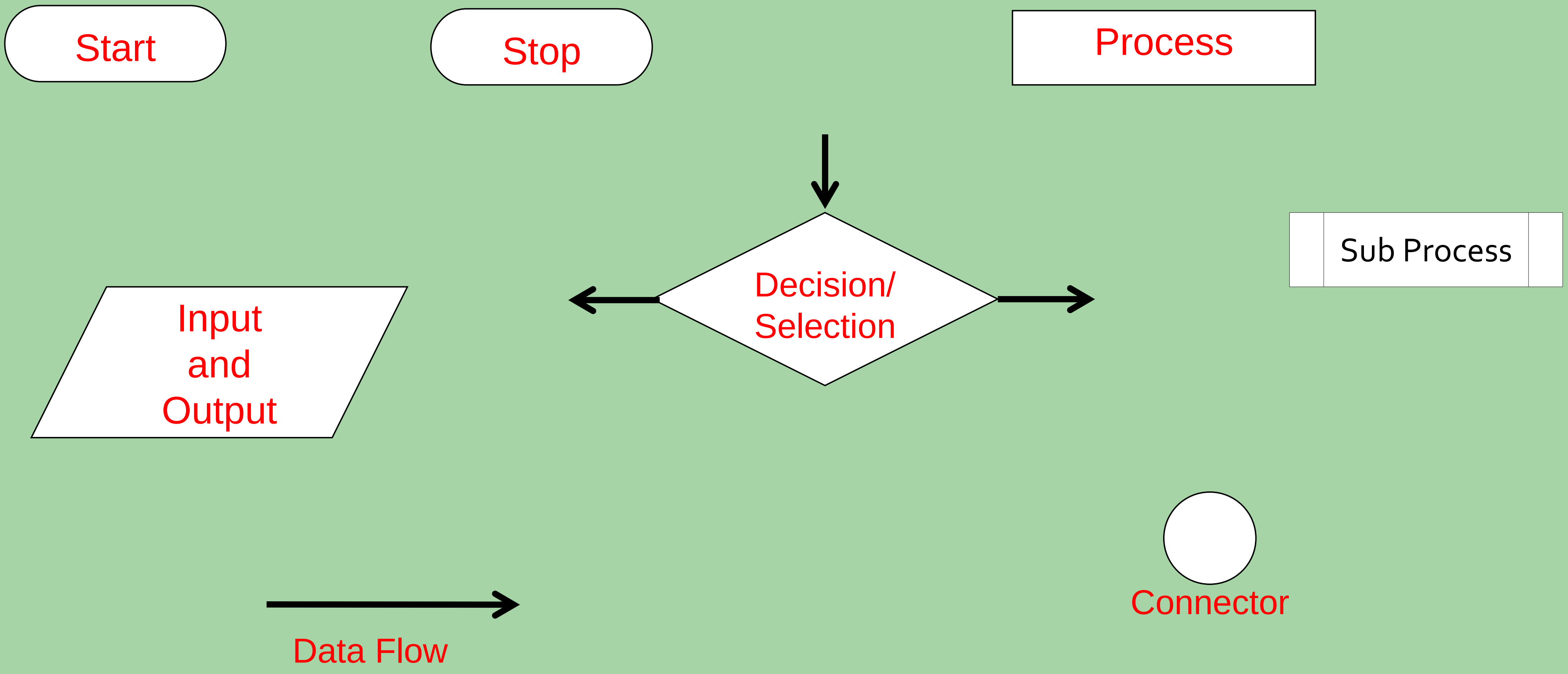
ක්‍රමලේඛ සංවර්ධන ක්‍රියාවලිය

1. ඇල්ගොරිදමයක් ගොඩ නැගීම
2. කේතනය
3. පරීක්ෂාකිරීම

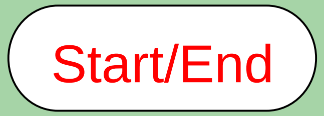
Flowcharts

- Flowcharts are examples for algorithms
- Flowcharts aid in breaking down problems because they only allow one step per symbol.
- Very good tool for break down the solution for simplest steps
- Using standard symbols for drawing flowcharts.
- Each symbol in a flowchart represents one step in the solution.
- Flowcharts are visual representations of an ***algorithm***.
- Flowcharts use easy-to-understand symbols to represent actions on data and the flow of data.
- Language independent.

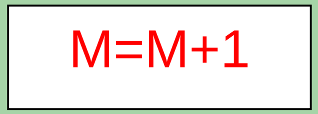
Symbols of the flowcharts



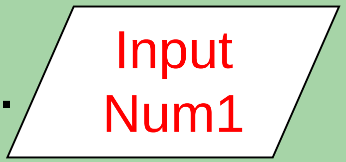
- **Terminators** begin and end each flowchart.



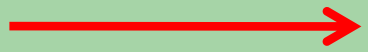
- A **Process** is used to represent a single event which requires no interaction from an outside entity.



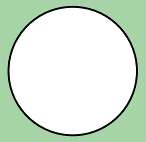
- **Input and Output** boxes show interaction with an outside entity.



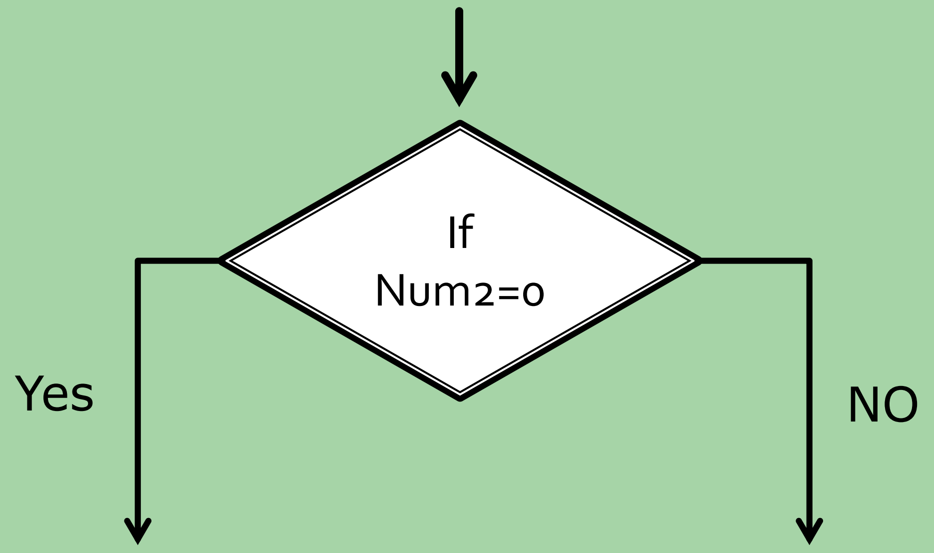
- **Data Flows** connect the different symbols.

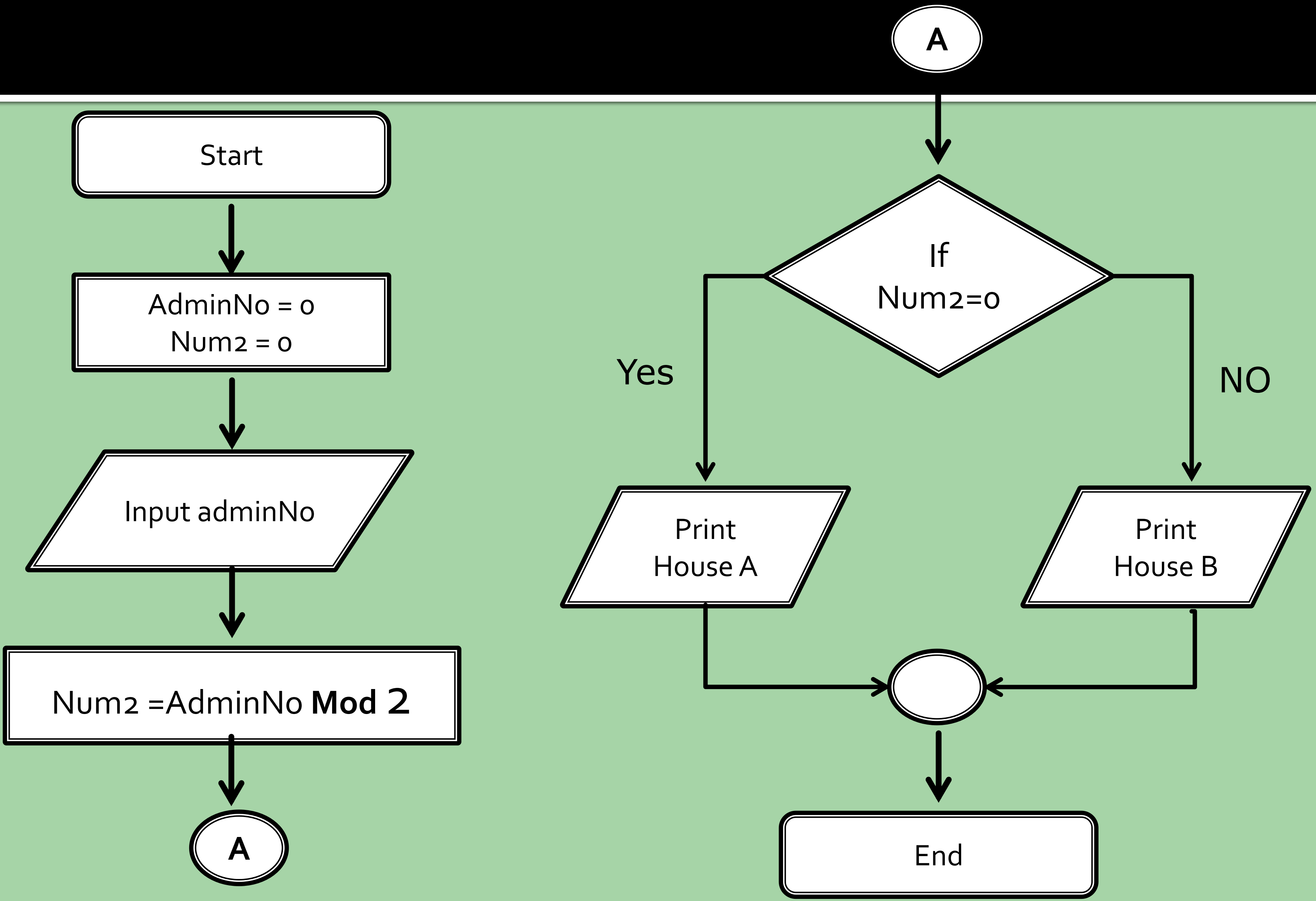


- **Connectors** can also be numbered, in order to allow movement from one page to another, for example.



- **Decision** Box create two data flows from one. A Decision diamond ALWAYS has True/False or Yes/No written on the left and right sides.




Start
AdminNo = 0
Num2 = 0
Input adminNo
Num2 =AdminNo **Mod** 2
A
A
If
Num2=0
Yes
NO
Print
House A
Print
House B
End

විචල්‍යයන් - Variables

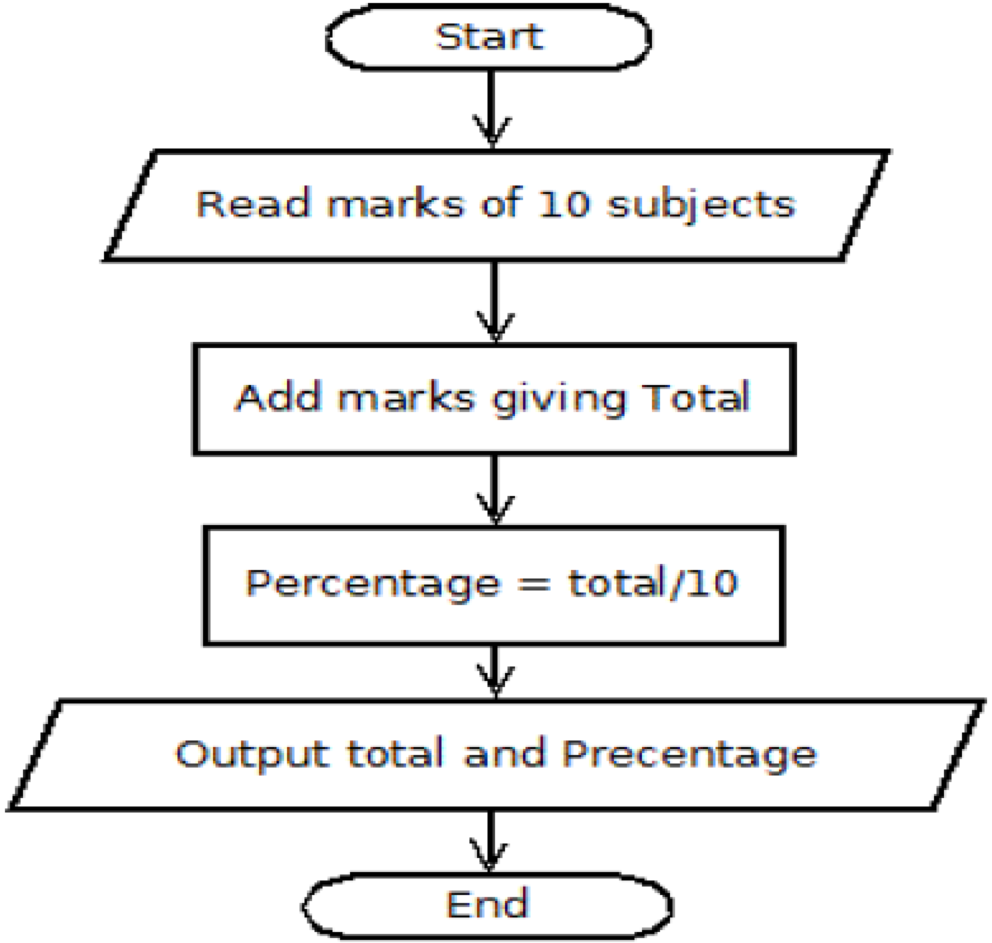
- විචල්‍යක් යනු යම් දත්තයක් ගබඩා කිරීම සඳහා පරිගණක මතකයෙහි වෙන්කර නම් කරන ලද ස්ථානයක් ය.

ක්‍රමලේඛ ක්‍රියාකරවීමේ දී දත්තයන් තාවකාලික ව ප්‍රධාන මතකයේ තබා ගැනීමට අවශ්‍ය වේ. මෙසේ දත්තයන් මතකයේ තාවකාලික ව තබා ගැනීම සඳහා විචල්‍යය භාවිත වෙයි. පරිගණකයේ ප්‍රධාන මතකයේ නිශ්චිත කොටසකට ප්‍රවේශ වීම සඳහා උපයෝගී කරගත හැකි සංකේත නාමයක් ලෙස විචල්‍යයක් දැක්විය හැකි ය. මෙලෙස විචල්‍යයක් මතකයේ කිසියම් කොටසකට තාවකාලික ව අනුබද්ධ කළ විට එම විචල්‍යය උපයෝගී කර ගෙන එම අදාළ මතක කොටසේ විවිධ දත්ත තාවකාලික ව ගබඩා කිරීම සහ නැවත ලබා ගැනීම සිදු කළ හැකි ය.

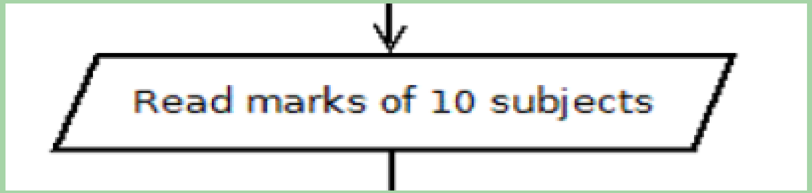
We can use comparison following operation in pythone

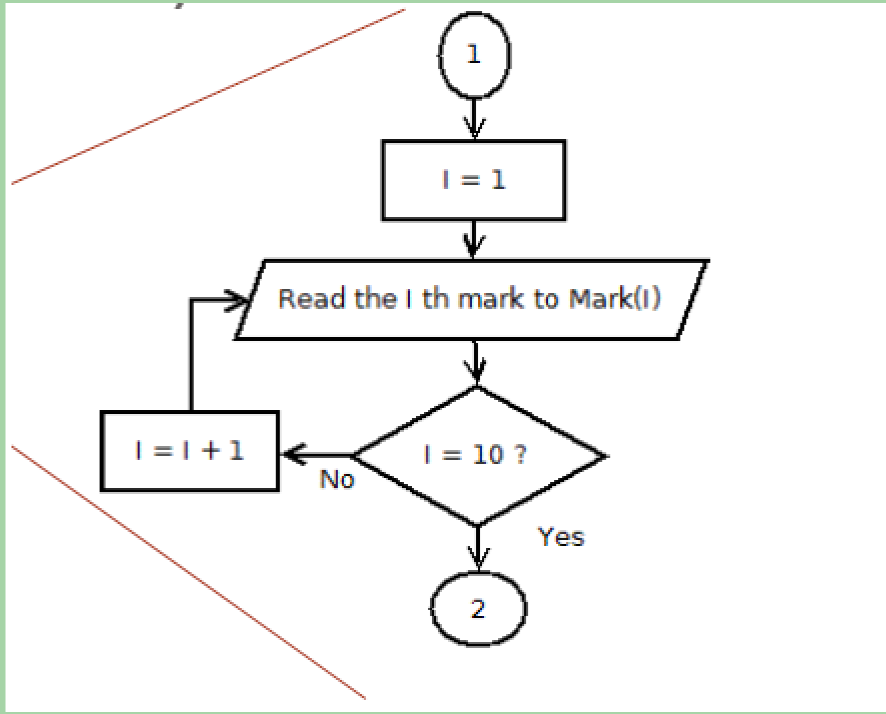
Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal

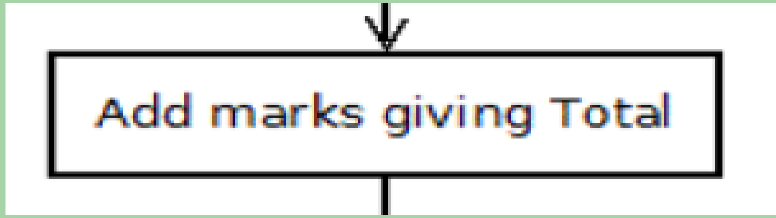
Example (Macro Level – Solution 1)

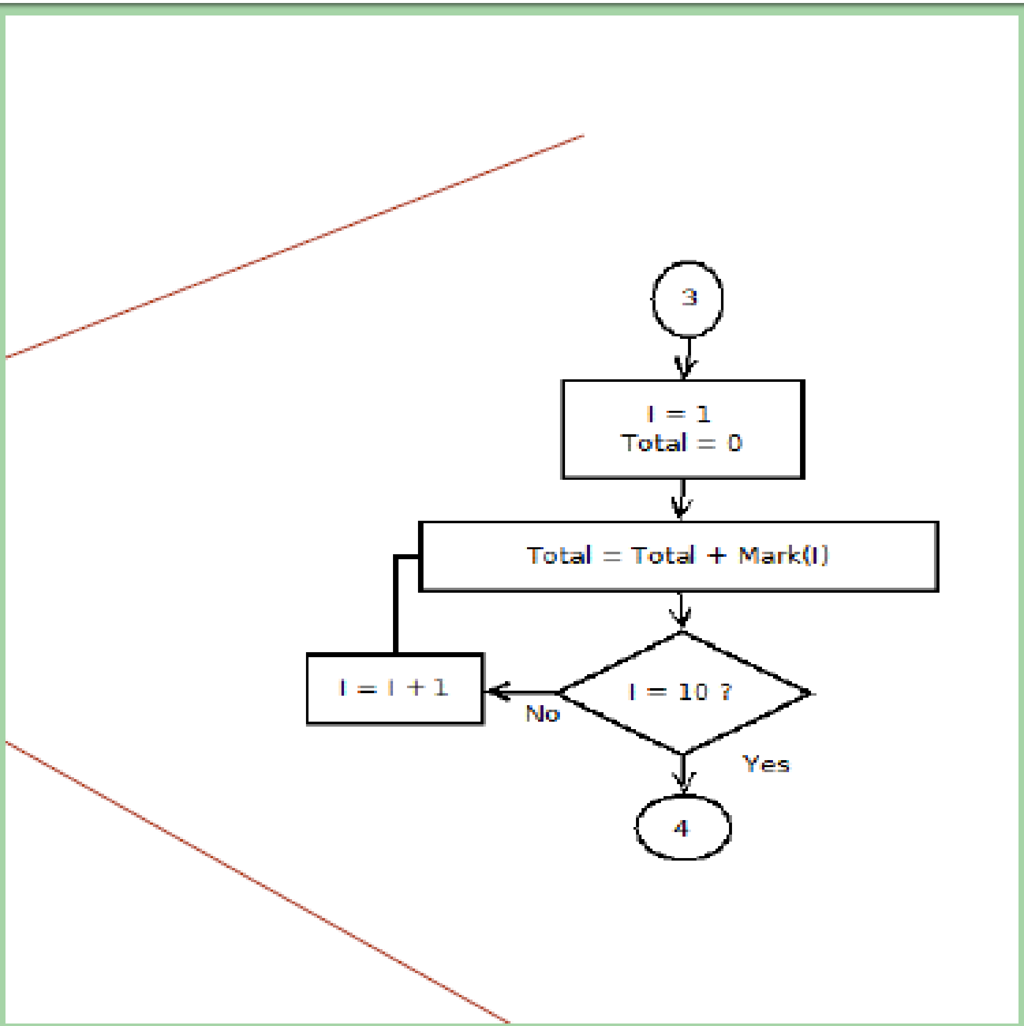


Micro Level


Read marks of 10 subjects


1
I = 1
Read the I th mark to Mark(I)
I = I + 1
No
I = 10 ?
Yes
2


Add marks giving Total

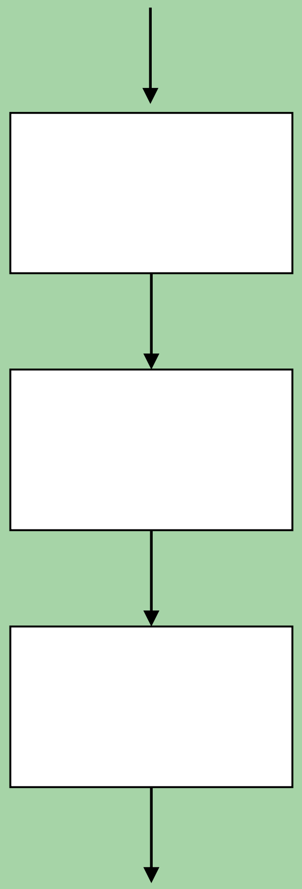

3
I = 1
Total = 0
Total = Total + Mark(I)
I = I + 1
No
I = 10 ?
Yes
4

Control Structures - පාලන ව්‍යුහයන්

ගැලීම් සටහන් සඳහා භාවිතා කරන පාලන ව්‍යුහයන් (Constructs) ආකාර 3කි.

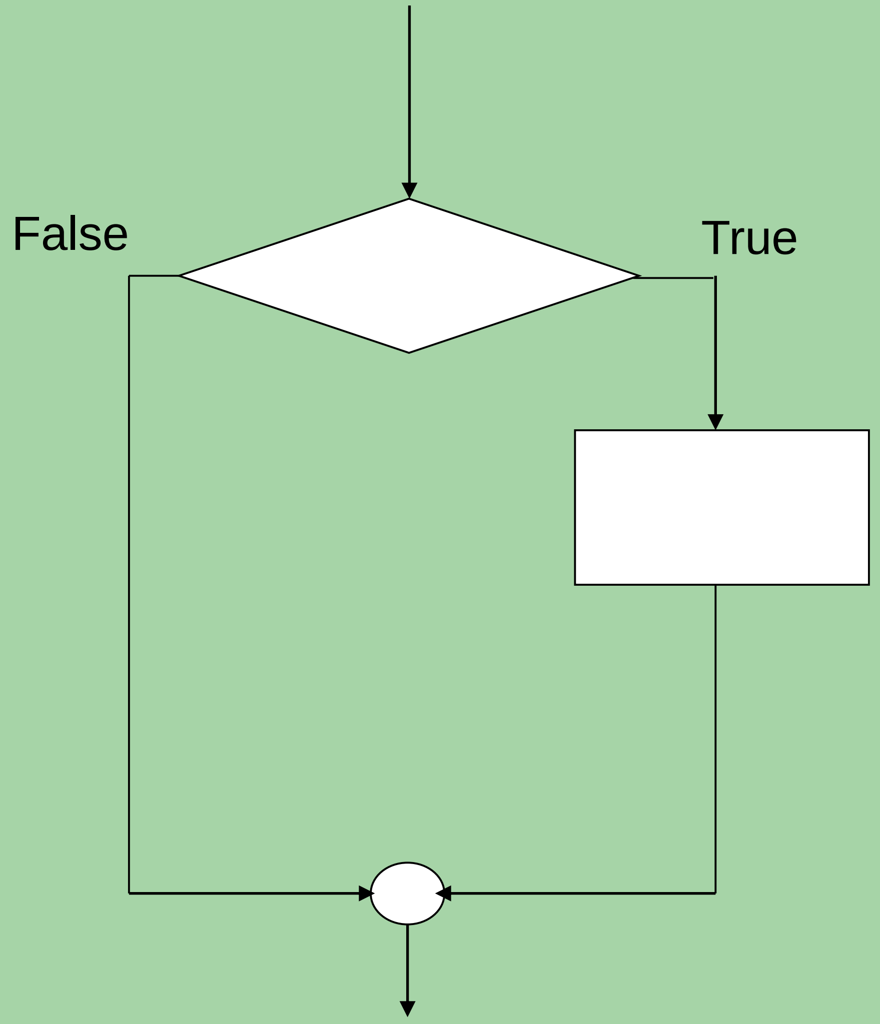
1. *Sequence – අනුක්‍රමය*
2. *Selection - තේරීම/වරණය*
3. *Iteration - පුණර්කරණය*

Sequence



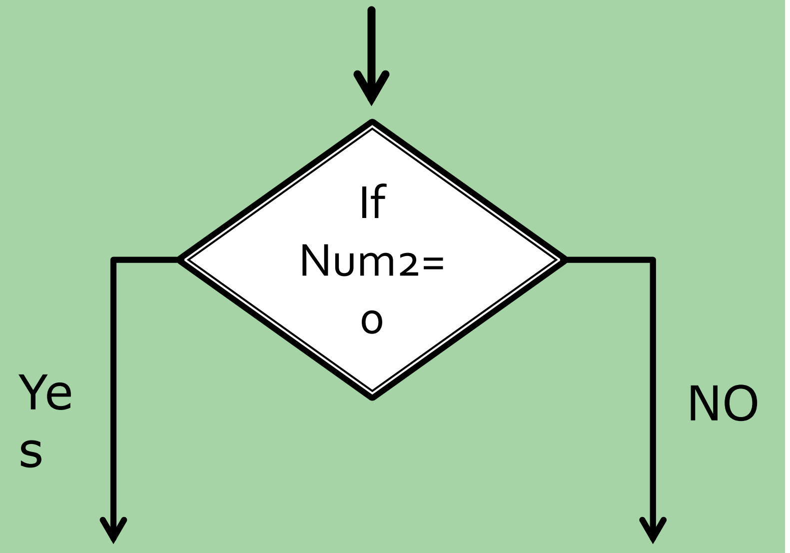
Sequence simply means to perform one step, then the next step, then the next step.

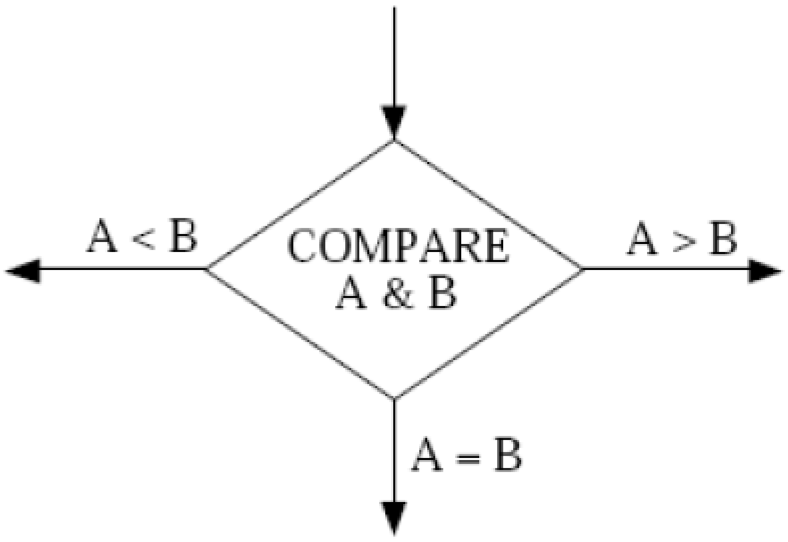
Selection - තේරීම/වරණය



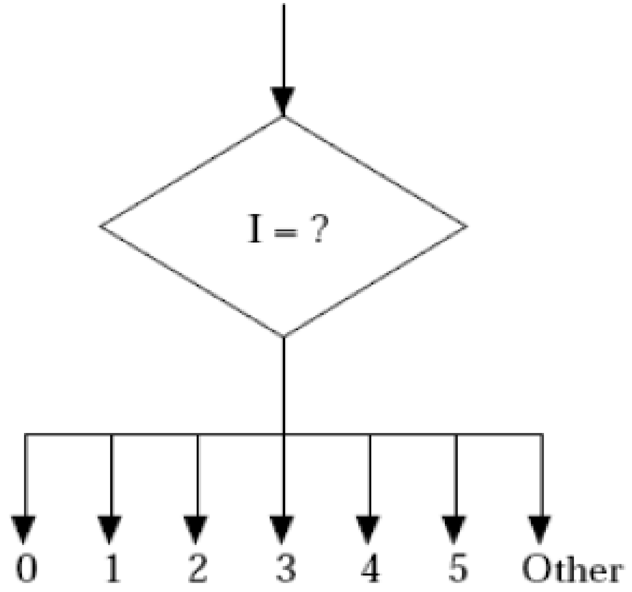
Selection means to perform an action only if a condition is true.

Simple Selection and Multiple Selection



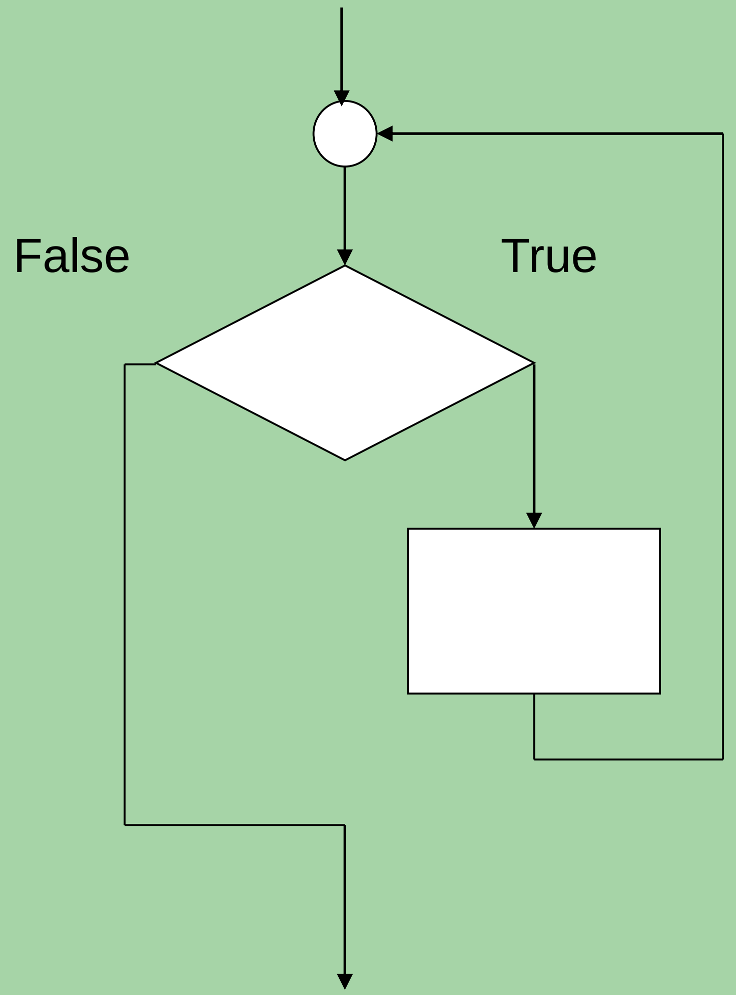


(b) Three-way branch



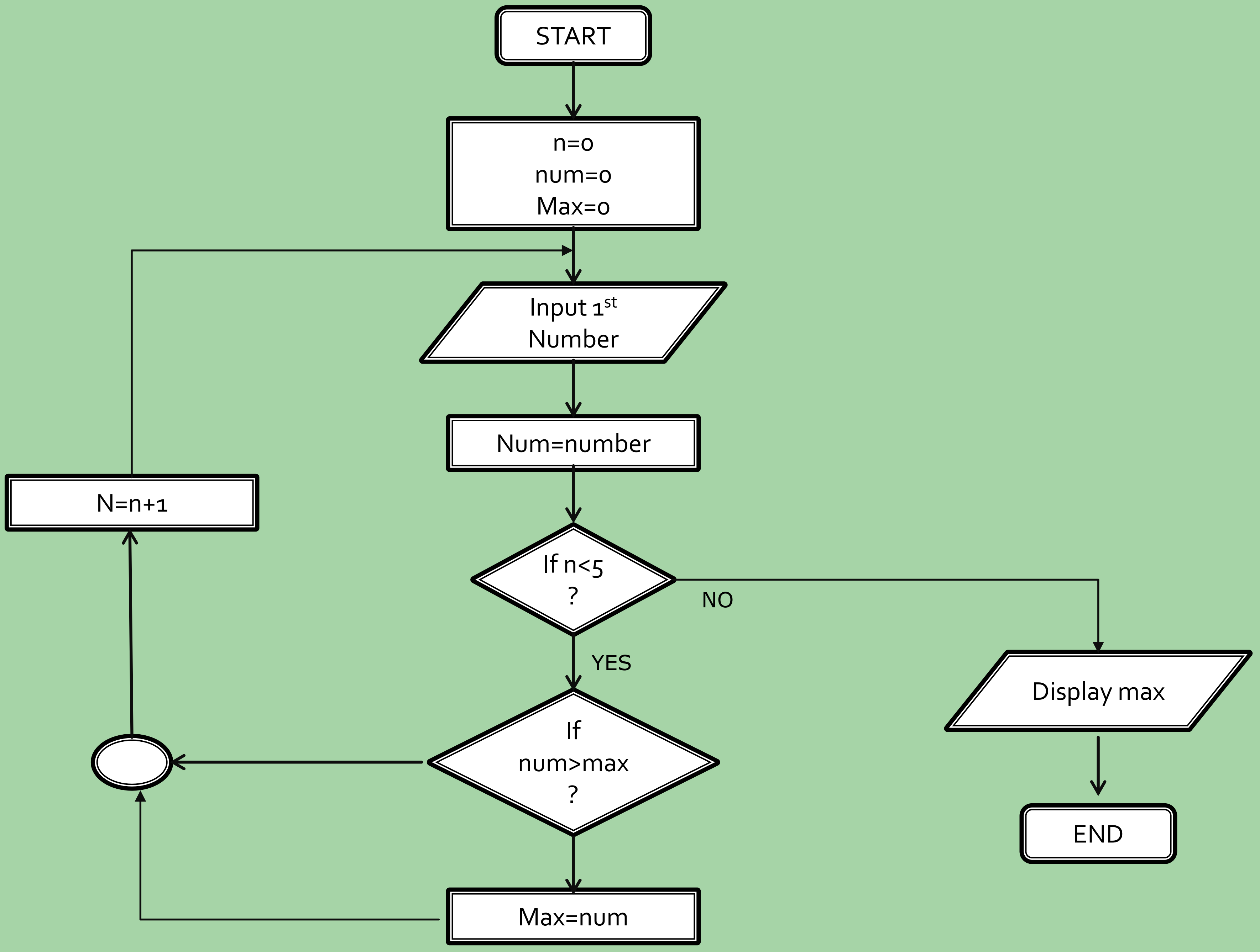
(c) Multiple-way branch

Repetition- පුණර්කරණය

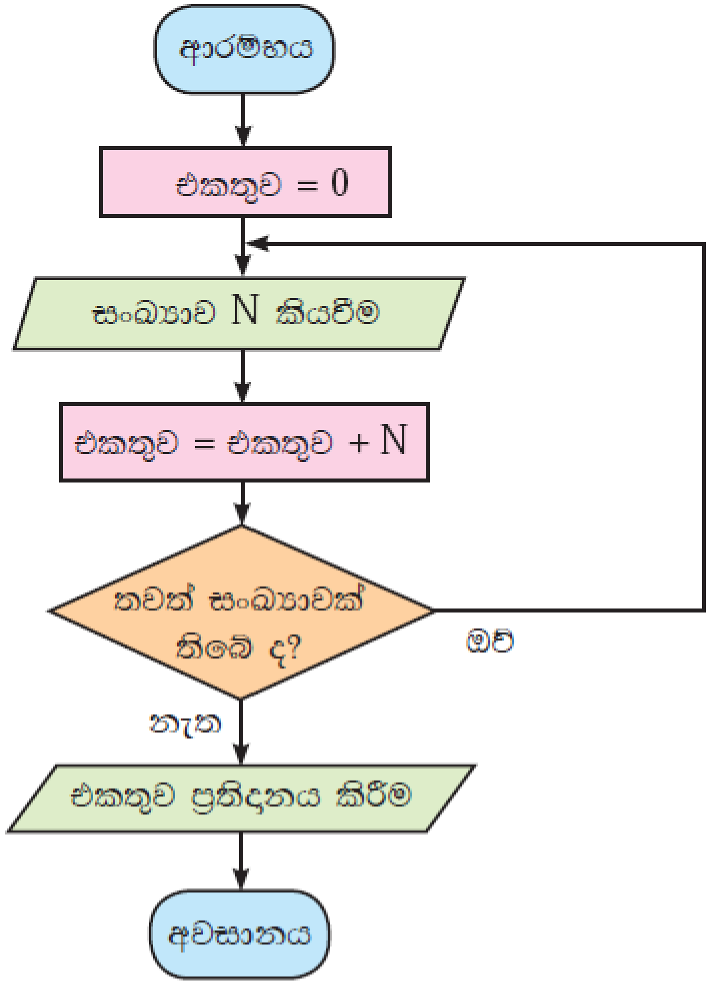


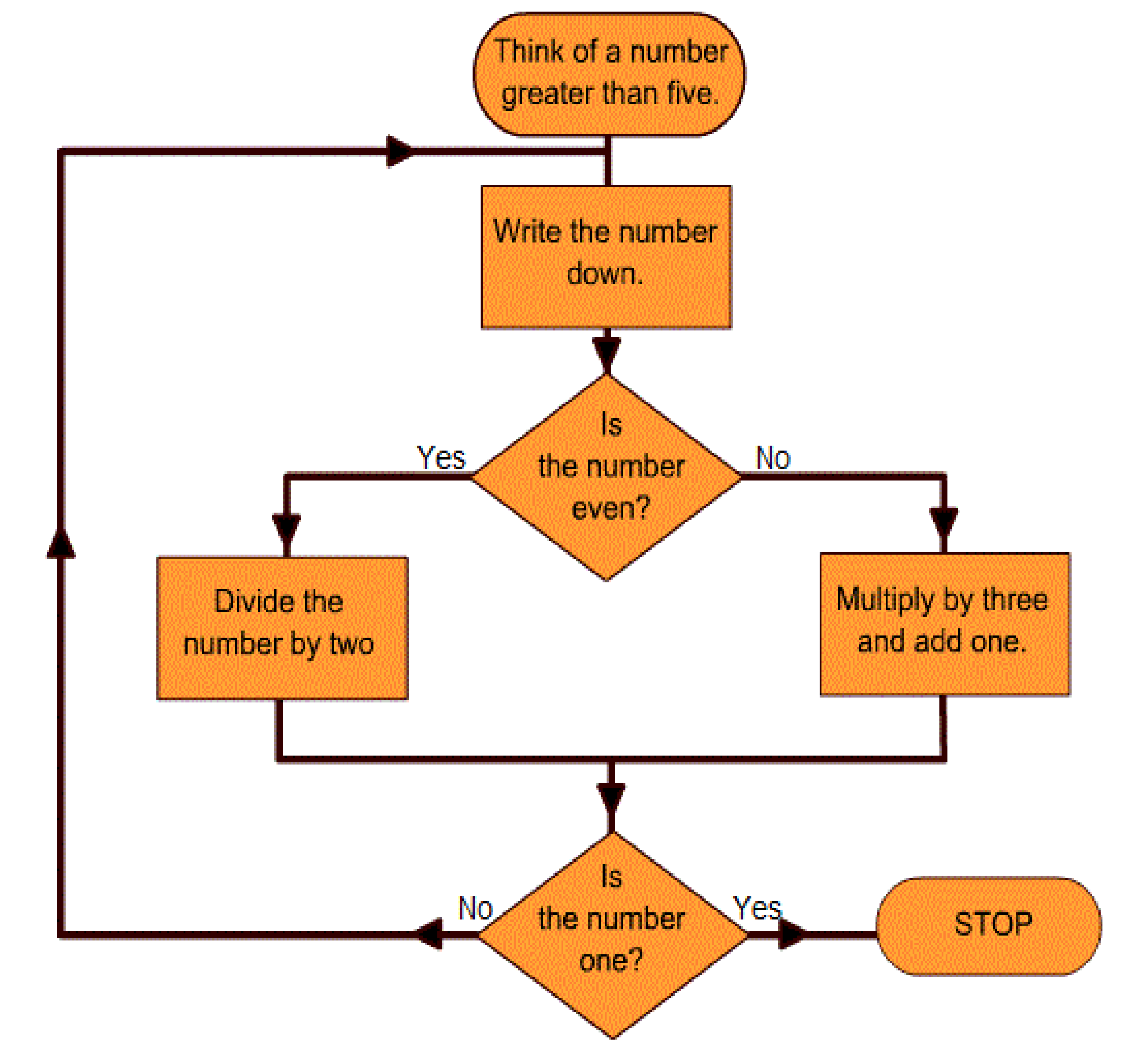
Iteration means to perform an action as long as a condition is true. It’s also known as ***looping.***

ඔබේ නම 5 වතාවක් තිරයේ මුද්‍රණය කිරීම


START
n=o
num=o
Max=o
Input 1st
Number
Num=number
N=n+1
If n<5
?
NO
YES
Display max
If
num>max
?
END
Max=num

සංඛ්‍යා සමූහයක එකතුව සෙවීම




Think of a number greater than five.
Write the number down.
Is the number even?
Yes
No
Divide the number by two
Multiply by three and add one.
Is the number one?
No
Yes
STOP

ගැලීම් සටහන් අඳින්න

1. සංඛ්‍යා 5කින් විශාලම සංඛ්‍යාව සෙවීම
2. සංඛ්‍යා 5කින් කුඩාම සංඛ්‍යාව සෙවීම
3. ඔබේ නම 5 වතාවක් තිරයේ මුද්‍රණය කිරීම
4. 7 හි මුල් ගුණාකාර 12 ලබා ගැනීම
5. පාඨමාලාවක් සමත්වීම සඳහා පැමිණීම 74% ට වැඩි වීම හා සාමාන්‍ය ලකුණු 59ට වඩා වැඩි විය යුතුය. සාමාන්‍ය ලකුණු 60 ට වඩාඅඩු නම් නැවත විභාගයට පෙනී සිටිය යුතු අතර පැමිණීම 75%ට වඩා අඩුනම් මුල සිට පාඨමාලාව හැදෑරිය යුතුය.

Problems

6. ළමුන් දස දෙනෙකු ගණිතය විෂයට ලබාගත් ලකුණු වල එකතුව හා සාමාන්‍ය අගය සොයා සාමාන්‍ය අගය 50 හෝ 50ට වැඩිනම් “Good” ලෙසත් නැතිනම් “Bad” ලෙසත් ප්‍රදර්ශණය කිරීමට ගැලීම් සටහනක් අඳින්න.

7. Draw a flowchart to find the sum of first 50 natural numbers.

8. Draw a flowchart for computing factorial N (N!) (එකේ සිට දෙන ලද සංඛ්‍යාවක් දක්වා සංඛ්‍යාවල ගුණිතය ලබා ගැනීම සඳහා ගැලීම් සටහනක් අඳින්න.)

Problems

9. දෙන ලද විෂයන් සංඛ්‍යාවක ලකුණු ඇතුලත් කල පසු එම ලකුණු වල එකතුව හා සාමාන්‍ය ලබා ගැනීම සඳහා ගැලීම් සටහනක් අඳින්න.
10. දෙන ලද සංඛ්‍යා රාශියකින් විශාලම සංඛ්‍යාව සෙවීම සඳහා ගැලීම් සටහනක් අඳින්න. (-1 ඇතුලත් කල විට වැඩසටහනින් ඉවත්වී විශාලම සංඛ්‍යාව මුද්‍රණය වේ.)
11. දෙන ලද සංඛ්‍යාවක් ඉතිරි නැතිව බෙදෙන සංඛ්‍යා සියල්ල මුද්‍රණය සඳහා ගැලීම් සටහනක් අඳින්න.

Pseudo Codes

ව්‍යාජ කේත

Pseudo Code

- පරිගණක වැඩසටහන් ලිවීමේ දී භාවිතා කරන පරිගණක භාෂාවේ නීති රීති වලට අනුකූලව එහි අඩංගු Keywords භාවිතා කර codes ලිවිය යුතුයි. මෙය පරිගණක භාෂාවෙන් භාෂාවට වෙනස් වේ.
- නමුත් මෙලෙස පරිගණක භාෂාවක් මත පදනම් නොවී යම් කාර්යක් ඉටු කිරීම සඳහා මානව භාෂාවක් භාවිතා කරමින් ලියන උපදෙස් (වගන්ති) ව්‍යාජ කේත ලෙස හඳුන්වයි. එසේ වුවද ව්‍යාජ කේත ලිවීමේ දී අත්‍යාවශ්‍යය Keywords කිහිපයක් භාවිතා කරයි.

Examples:- BEGIN, START, END

SET, CALCULATE, COMPUTE

INPUT, READ, GET, OBTAIN

OUTPUT, PRINT, DISPLAY

IF THEN ELSE END IF, WHILE END WHILE, FOR NEXT, REPEAT UNTIL

Rules for Pseudo codes

එක පේළියක එක වගන්තියක් පමණක් ලියන්න

Keywords කැපිටල් අකුරෙන් ලියන්න (Not Essential)

අතු බෙදීම් පෙන්වීමට Indents තබන්න

End multiline structures

Keep statements language independent

පාලන ව්‍යුහ - 1. අණුක්‍රමණය (Sequential)

BEGIN

Width = 0

Height = 0

Area = 0

INPUT width

INPUT HEIGHT

Area=width * Height

Display area

End

Assign Values for the Variables

Sequence

2. සරල තේරීම - Selection (IF-THEN-ELSE)

Binary choice on a given Boolean condition is indicated by the use of four keywords: IF, THEN, ELSE, and ENDIF. The general form is:

```
IF condition THEN
        sequence 1
ELSE
        sequence 2
ENDIF
```

තෝරා ගැනීමේ ප්‍රකාශන (Selection Statements)

IF Then ප්‍රකාශනය

උදාහරණ

```
        If marks >= 50 Then
            Display "Pass"
        Else
            Display "Fail"
        Endif
```

Selection ((IF-THEN-ELSE)

```
Begin
Input number1
Input number2
If number1>number2 Then
    Print number1
    Print this is larger one
Else
    Print number2
    Print this is larger one
End if
End
```

පුනරාවර්තක ප්‍රකාශන - Repetitive/Iteration Statements

1. For Next
2. While Do
3. Repeat Until

WHILE DO Loop

The WHILE construct is used to specify a loop with a test at the top.

The beginning and ending of the loop are indicated by two keywords WHILE and ENDWHILE.

The general form is:

```
WHILE condition DO
        Statements
        loop control statement
ENDWHILE
```

Variable Declaration

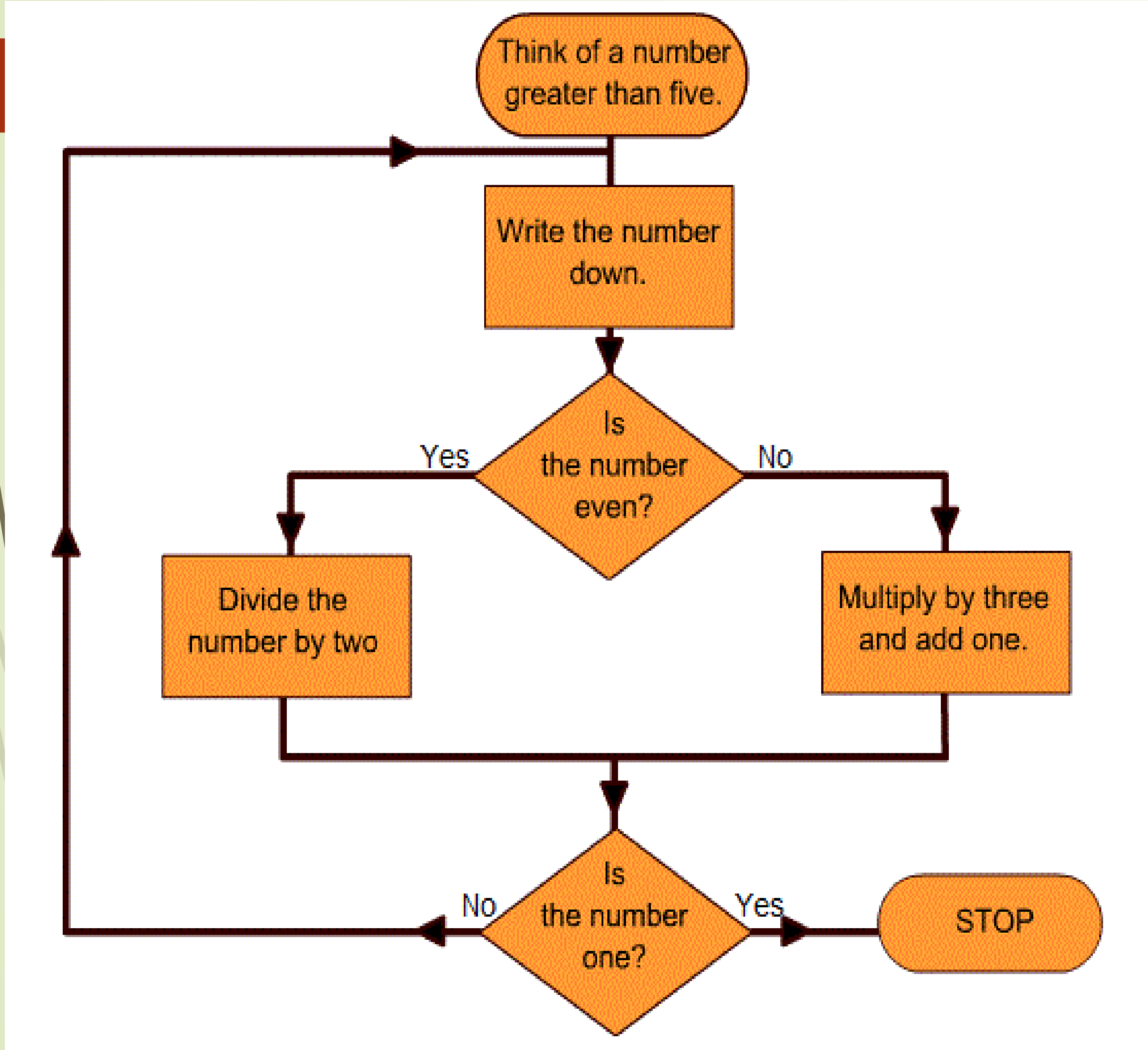
Condition

Control Statement

```
BEGIN
Total=0
Count=0
While count<5 Do
    INPUT num
    total=total+num
    count=count+1
ENDWHILE
PRINT total
END
```

We can use WHILE DO Loop without control statement mention below.

```
BEGIN
Total=0
Num=0
While num>=0 Do
        INPUT num
        total=total+num
ENDWHILE
PRINT total
END
```



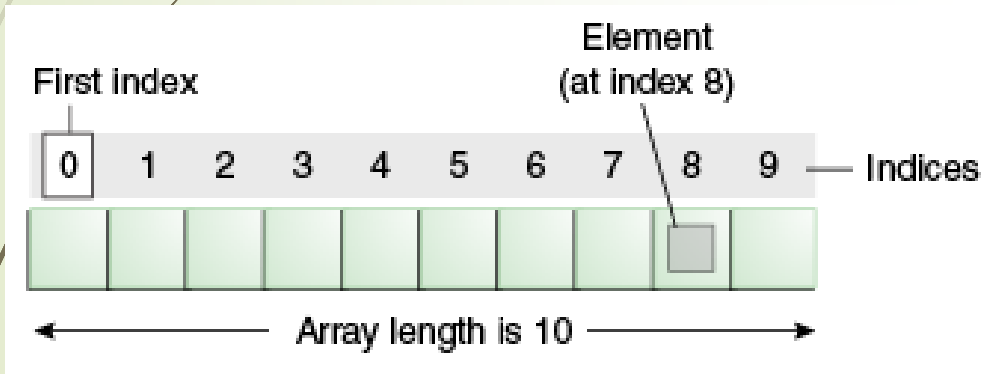

```
BEGIN
WRITE a number

REPEAT
        IF number even THEN
                n=number/2
        ELSE
                n=number*3+1
        END IF

UNTIL n=1
END
```

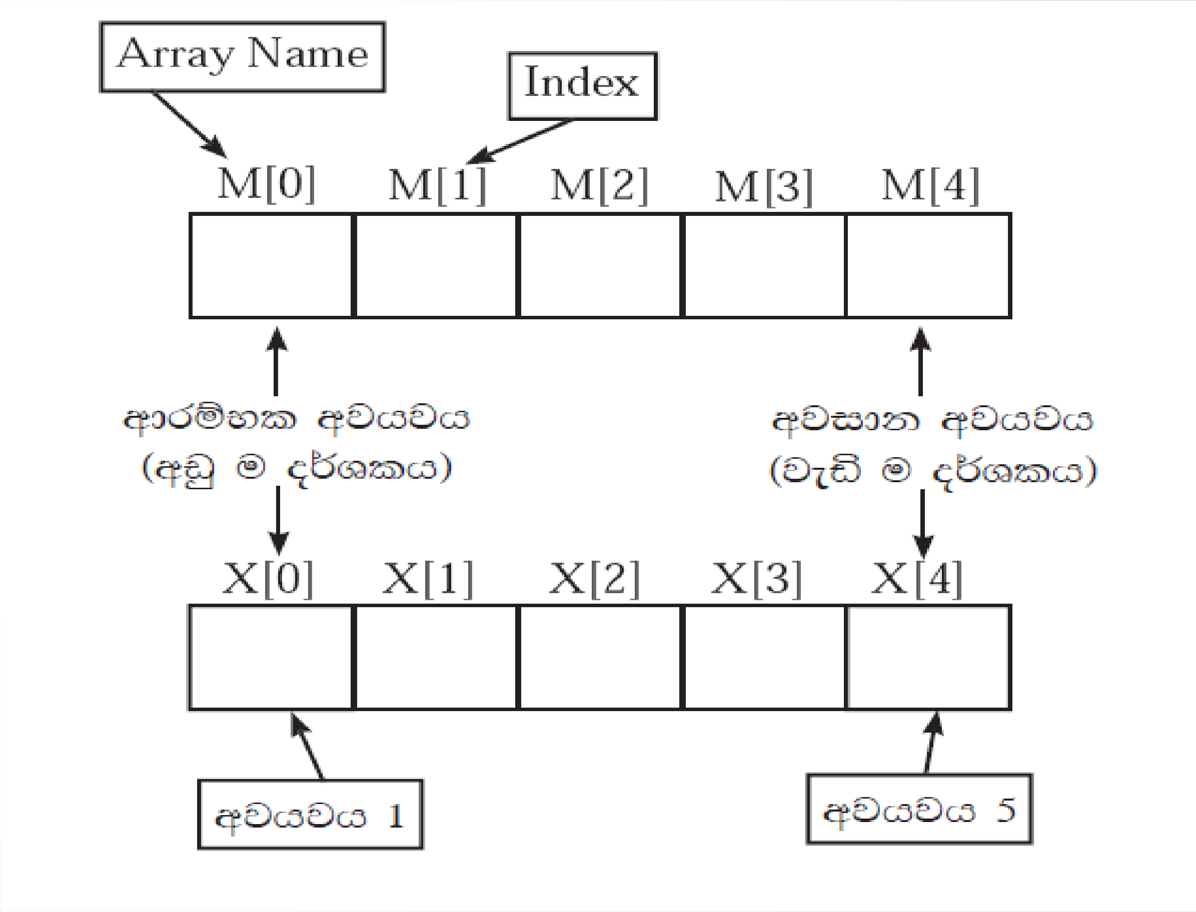
Constant

- In computer programming, a constant is a value that cannot be altered by the program during normal execution,

Arrays

- එකම ප්‍රරූපයකට අයත් අසමාන අගයන් සහිත දත්ත එක් විචල්‍ය නාමයක් යටතේ ගබඩා කිරීම අරාවන් (Arrays) ලෙස හඳුන්වයි.





REPEAT UNTIL Loop

This loop is similar to the WHILE loop except that the test is performed at the bottom of the loop instead of at the top. Two keywords, REPEAT and UNTIL are used. The general form is:

```
REPEAT
      Sequence
      loop control statement
UNTIL condition
```

```
Begin
i=0
Total =0
Repeat
    Total=total+1
    i=i+1
Until i=10
Print total
End
```

Control Statement and condition?

i=10 ලෙස ගෙන ඉහත ප්‍රතිදානයම ලැබීමට ව්‍යාජ කේතය නැවත ලියන්න

FOR Loop

This loop is a specialized construct for iterating a specific number of times, often called a "counting" loop. Two keywords, FOR and NEXT are used. The general form is:

```
FOR criteria
        Statesment1
        Statesment2
        ..................
Next
```

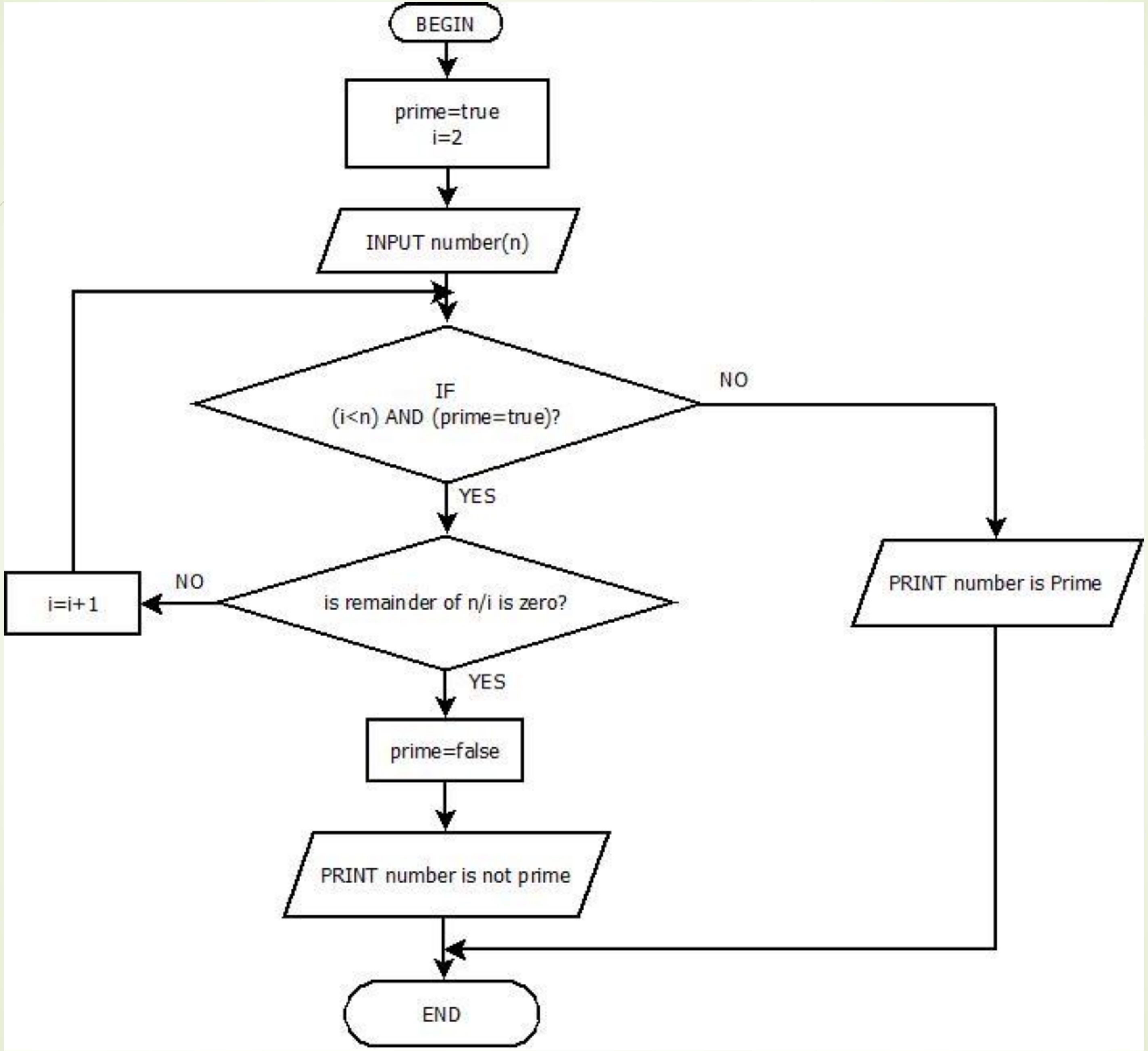
```
BEGIN
Total = 0
FOR i=1 to 10
    total =total +1
NEXT i
PRINT total
END
```

```
Begin
For (counter=1 to 10)
    Print counter
Next counter
End
```

```
Begin
For (counter=1 to 10 Step 2)
    Print counter
Next counter
End
```

Check prime Numbers


```
Begin
    prime=true
    READ a number
    Count=2
    WHILE (count<number) and (prime=true) DO
      IF number is divisible by count THEN
         prime=false
      ELSE
        count=count+1
      END IF
    ENDWHILE
    If prime=true then
       display number "is prime"
    Else
        display number "is not prime"
    End if
End
```


BEGIN
prime=true
i=2
INPUT number(n)
IF
(i<n) AND (prime=true)?
NO
YES
is remainder of n/i is zero?
NO
i=i+1
YES
prime=false
PRINT number is not prime
PRINT number is Prime
END

```
Begin
    Read number
    Max=number
    Min=number
    While number >=0
        if number>max then
                max=number
       else if  number<min then
               min=number
     end if
     read number
    end while
    Display "Maximum is " max
    Display "Minimum is " min
End
```

Nested Iteration / නීඩිත පුනර්කරණය

```
Begin
line=1
        While line<=10
        stars=1
                while stars<=line
                        display "*" on the same line
                        stars=stars+1
                end while
        line=line+1
        Go to a new line
        End while
End
```

හස්තානුරේඛන (Hand Traces)

Hand-tracing is a simulation of code execution in which you step through instructions and track the values of the variables.

ක්‍රමලේඛ සංවර්ධන ක්‍රියාවලියේ (ගැටළු විසඳීමේ ක්‍රියාවලියේ) පියවර

1. ගැටළුව තේරුම් ගැනීම/අර්ථ දැක්වීම
2. විසඳුම සැළසුම් කිරීම
3. ඇල්ගොරිදම සංවර්ධනය
4. ඇල්ගොරිදම පරීක්ෂා කිරීම
5. ඇල්ගොරිදමය ක්‍රමලේඛයක් බවට කේතනය කිරීම
6. ක්‍රමලේකය පරික්ෂාකිරීම හා නිදොස් කිරීම
7. ක්‍රමලේකය නඩත්තුව හා ලේඛනගත කිරීම

Modularization

- Program එකක් විශාල සංකීර්ණ මෘදුකාංගයක් විවිධ කේත කොටස් වලින් සමන්විත විය හැකිය. කේත මිලියන ගණනක් (millions of lines of code) සමඟ programmers ල සිය ගණනකට වැඩ කිරීමට සිදුවේ. මෙවැනි පරිසරයක දී කේත මග හැරීම, මෙන්ම එකම කේත කොටස් නැවත නැවත ලිවීමට ද සිදු වේ.

- මෙම ගැටළු වලට විසඳුමක් ලෙස "modularization," යොදා ගනී.

Modularization

- විශාල පරිගණක ක්‍රමලේඛයක් මොඩියුල වලට වෙන්කර ලිවීම මෙමගින් සිදුකරයි.
- **Module:**
  - පරිගණක වැඩසටහනක් කාර්යයන් සමූහයක එකතුවකි. එවැනි පරිගණක වැඩසටහනක සමස්ථ කාර්යය වෙන් වෙන් වශයෙන් කාර්යයන් වලට වෙන්කර ගැනීමෙන් සමස්ථ කාර්යයම පහසු වේ. වෙන්කර ගත් එවැනි අනු කාර්යක් ඉටු කිරීම සඳහා වෙන්ව ලියන ලද කේත (codes) වල එකතුවක් තනි ඒකකයක් ලෙස ගත්කල Module එකක් ලෙස හැඳින්විය හැකිය.
  - විවිධ පරිගණක භාෂා වල මෙම modules විවිධ නම් වලින් හඳුන්වයි.
    - Examples: subroutine, procedure, function, or method
  - උදා:-සෙල්සියස් වලින් උෂ්ණත්වය ආදානය කල විට ෆැරන්හයිට් වලට පරිවර්තනය කර ප්‍රතිදානය ලබා දීම.

    Module 1. Input Procedure, Module 2:- Process Function, Module 3:- Output Procedure

- Modularization මගින්,

1. Codes becomes reusable - කේත නැවත නැවත භාවිතා කල හැකිය
2. Easier to debug - දෝෂ හඳුනා ගැනීම පහසුය
3. Easier to manage - ක්‍රමලේඛනය කිරීම කළමණාකරනය පහසුය
4. Allows multiple programmers to work simultaneously - එක විට Programmers වැඩි පිරිසකට programming කල හැකිය. එක් එක් මොඩියුලය වෙන වෙනම නිමකල හැකිය
5. Makes identifying structures easier - සමස්ථ program එකේ ව්‍යුහය අවබෝධ කර ගැනීම පහසුය

Modularization

- Program එකක තිබිය හැකි මොඩියුල ගණන සීමා රහිත වේ.
- මොඩියුලයක් ක්‍රියාත්මක වීම (execute) සඳහා **Call(invoke)** කල යුතුයි.
- මොඩියුලයක් ක්‍රියාත්මක වීම (execute) සඳහා **Call(invoke)** කල හැකි වාර

  ගණන සීමා රහිත වේ.

Modularizing a Program

- බොහෝ programs ප්‍රධාන Program එකකින් (main) සමන්විත වන අතර, Main Program එක තුල අනෙකුත් Modules අඩංගු කරයි. එහි දී එම modules වෙන් වෙන්ව හඳුනා ගැනීම සඳහා නම් කල යුතුයි.

- Rules for naming modules different for every programming language
  - For this text:
    - Must be one word
    - Should be meaningful
    - Followed by a set of parentheses
    - Corresponds to module naming in Java, C++, C#

Modularizing a Program
Naming a Module

DON'T DO IT

These identifiers are not recommended.

Suggested Module Names	Comments
Superior identifiers	
`calculateGrossPay()`	Good
`calculateGross()`	Good—most people would interpret "Gross" to be short for "Gross pay"
Inferior or illegal identifiers	
`calGrPy()`	Legal, but cryptic
`calculateGrossPayForOneEmployee()`	Legal, but awkward
`calculate gross()`	Not legal—embedded space
`calculategrosspay()`	Legal, but hard to read without camel casing

Mashups

- සංකලනය යනු වෙබ් අඩවි සංවර්ධනයේ දී භාවිතා කරන සංකල්පයකි. වෙබ් පිටු හෝ වෙබ් යෙදුම් සකස් කිරීමේ දී ප්‍රභව එකකට වඩා වැඩි සංඛ්‍යාවකින් ලබා ගන්නා තොරතුරු එක් රූපමය අතුරු මුහුනතක ප්‍රදර්ශනය කිරීම මෙමගින් හඳුන්වයි.

ක්‍රමලේඛ සුසමාදර්ශ - programming paradigms

- ක්‍රමලේඛ සුසමාදර්ශයක් යනු ක්‍රමලේඛ කේත ලිවීමේ යම් රටාවක් හෝ ආකාරයක් හෝ විස්තර කරන ක්‍රමවේදයකි
- ක්‍රමලේඛ භාෂා වර්ගීකරණයට ද යොදා ගනී
- ප්‍රකට ක්‍රමලේඛ සුසමාදර්ශ
  - විධානාත්මක - imperative
  - ප්‍රකාශණාත්මක - declarative
  - වස්තු නැඹුරු - object Oriented
- සමහර පරිගණක භාෂා වලට සුසමාදර්ශ එකකට වඩා තිබිය හැකිය

  උදා:- python –imperative and object oriented

Good problem solving skills

Knowledge

Experience

Logic

End of the problem solution