Pseudocode

LO:

express a solution using standard design tools

Pseudocode - informal high-level description of a computer program or other algorithm, intended for human reading rather than machine reading.

Rules for Pseudocode

- Write only one statement per line
- Capitalize initial keyword
- Indent to show hierarchy
- End multiline structures
- Keep statements language independent

One Statement Per Line

Each statement in pseudocode should express just one action for the computer. If the task list is properly drawn, then in most cases each task will correspond to one line of pseudocode.

Task List

```
Read name, hours worked, rate of pay
Perform calculations
        gross = hours worked * rate of pay
Write name, hours worked, gross
```

Pseudocode

```
READ name, hoursWorked, payRate
gross = hoursWorked * payRate
WRITE name, hoursWorked, gross
```

Capitalize Initial Keyword

In the example below note the words: READ *and* WRITE. *These are just a few of the keywords to use, others include:*

READ, WRITE, IF, ELSE, ENDIF, WHILE, ENDWHILE

Pseudocode

```
READ name, hoursWorked, payRate
gross = hoursWorked * payRate
WRITE name, hoursWorked, gross
```

Indent to Show Hierarchy

Each design structure uses a particular indentation pattern

- **Sequence:**
  Keep statements in sequence all starting in the same column

- **Selection:**
  Indent statements that fall inside selection structure, but not the keywords that form the selection

- **Loop:**
  Indent statements that fall inside the loop but not keywords that form the loop

```
READ name, grossPay, taxes
IF taxes > 0
        net = grossPay – taxes
ELSE
        net = grossPay
ENDIF
WRITE name, net
```

End Multiline Structures

```
READ name, grossPay, taxes
IF taxes > 0
    net = grossPay – taxes
ELSE
    net = grossPay
ENDIF
WRITE name, net
```

See the IF/ELSE/ENDIF as constructed above, the ENDIF is in line with the IF.

The same applies for WHILE/ENDWHILE etc…

Language Independence

Resist the urge to write in whatever language you are most comfortable with, in the long run you will save time. Remember you are describing a logic plan to develop a program, you are not programming!

Defining Variables

```
DEFINE x AS integer
```

Variable Assignments

```
a ← 34
```

Totalling and Counting

```
x ← a + b
```

Input and Output

```
INPUT name
PRINT 'What is your name?‛
```

Selection

```
IF x = 1 THEN
   print "Hello«
ELSE IF x = 2 THEN
   print "How are you?«
ELSE
   print "Goodbye«
ENDIF
```

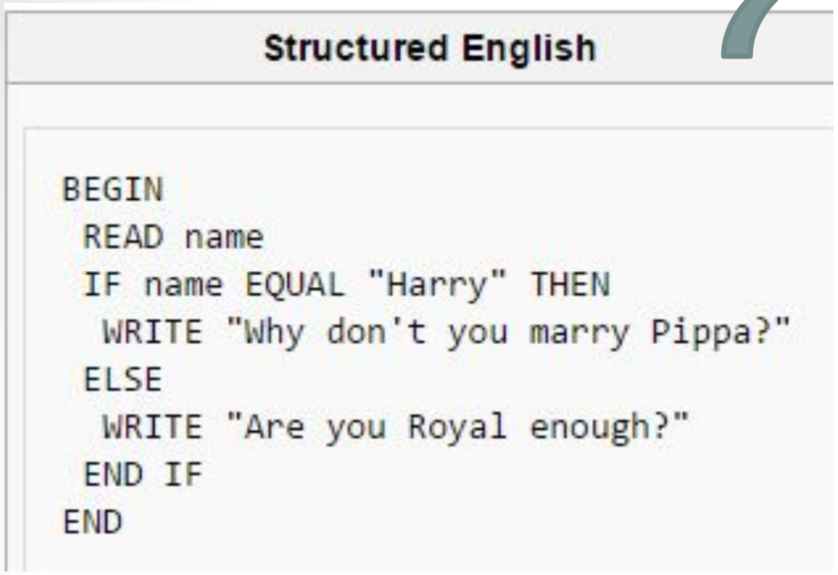
```
CASE x OF
   1 : PRINT "Hello«
   2 : PRINT "How are you?"
   3 : PRINT "I am fine"
   4 : PRINT "Have a good day!«
OTHERWISE
   PRINT "Goodbye"
ENDCASE
```

Iteration

```
FOR x = 1 TO 10
print x
NEXT
```

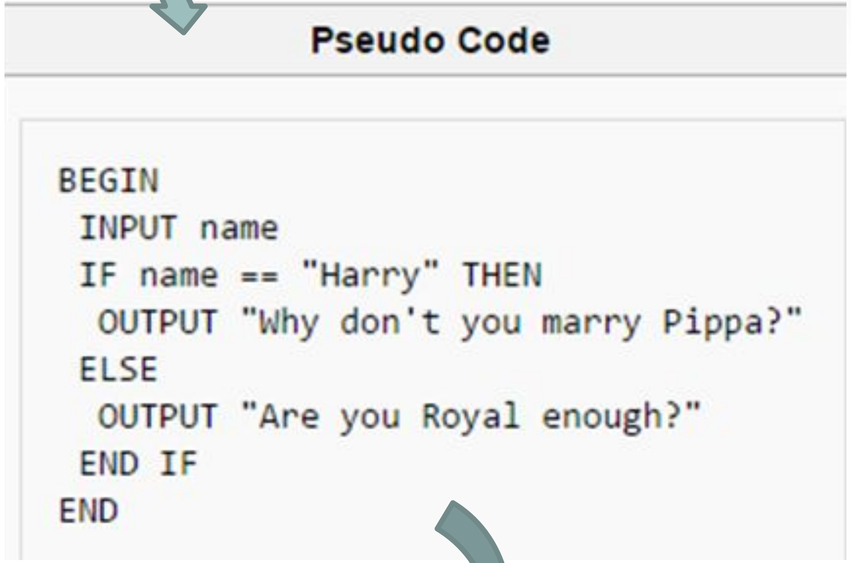
```
REPEAT
   INPUT x
UNTIL x < 10
```

```
INPUT x
WHILE x < 10
   INPUT x
ENDWHILE
```



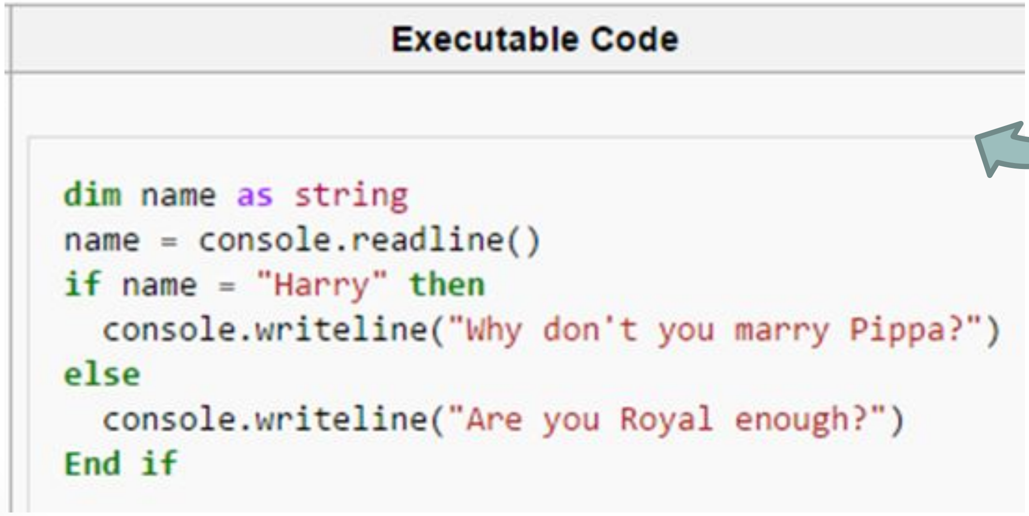
Structured English

```
BEGIN
 READ name
 IF name EQUAL "Harry" THEN
  WRITE "Why don't you marry Pippa?"
 ELSE
  WRITE "Are you Royal enough?"
 END IF
END
```



Pseudo Code

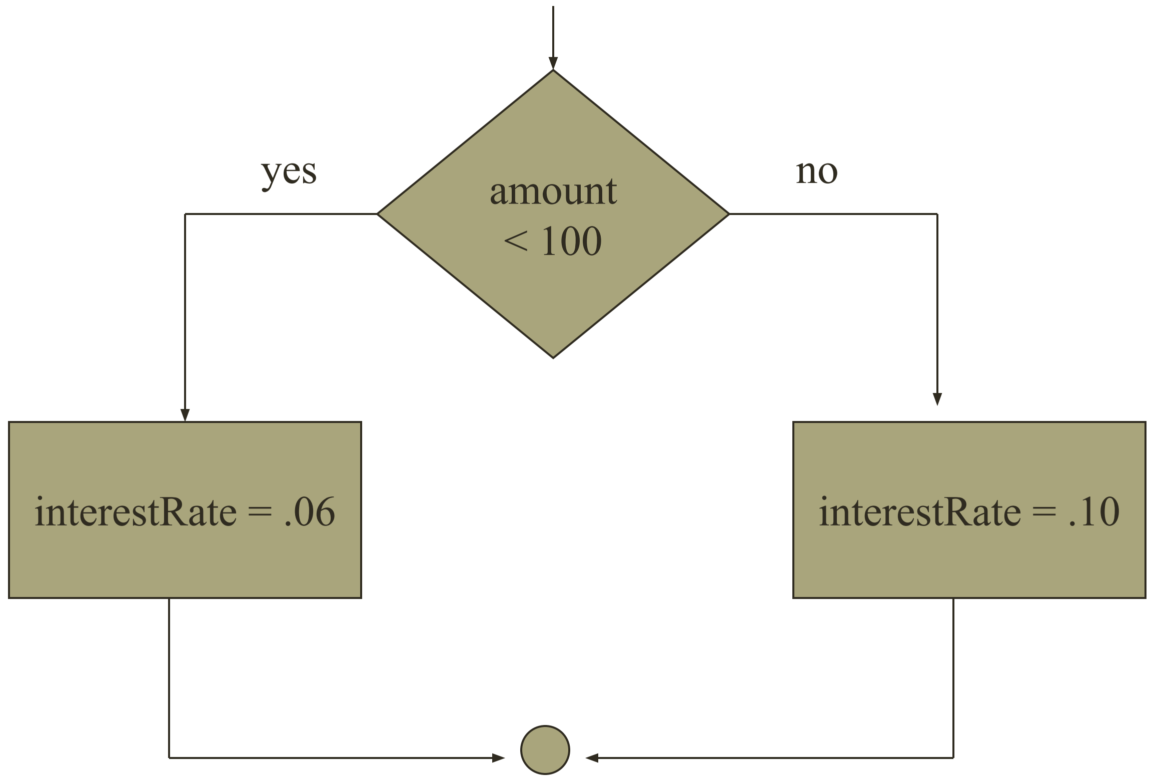
```
BEGIN
 INPUT name
 IF name == "Harry" THEN
  OUTPUT "Why don't you marry Pippa?"
 ELSE
  OUTPUT "Are you Royal enough?"
 END IF
END
```



Executable Code

```
dim name as string
name = console.readline()
if name = "Harry" then
  console.writeline("Why don't you marry Pippa?")
else
  console.writeline("Are you Royal enough?")
End if
```

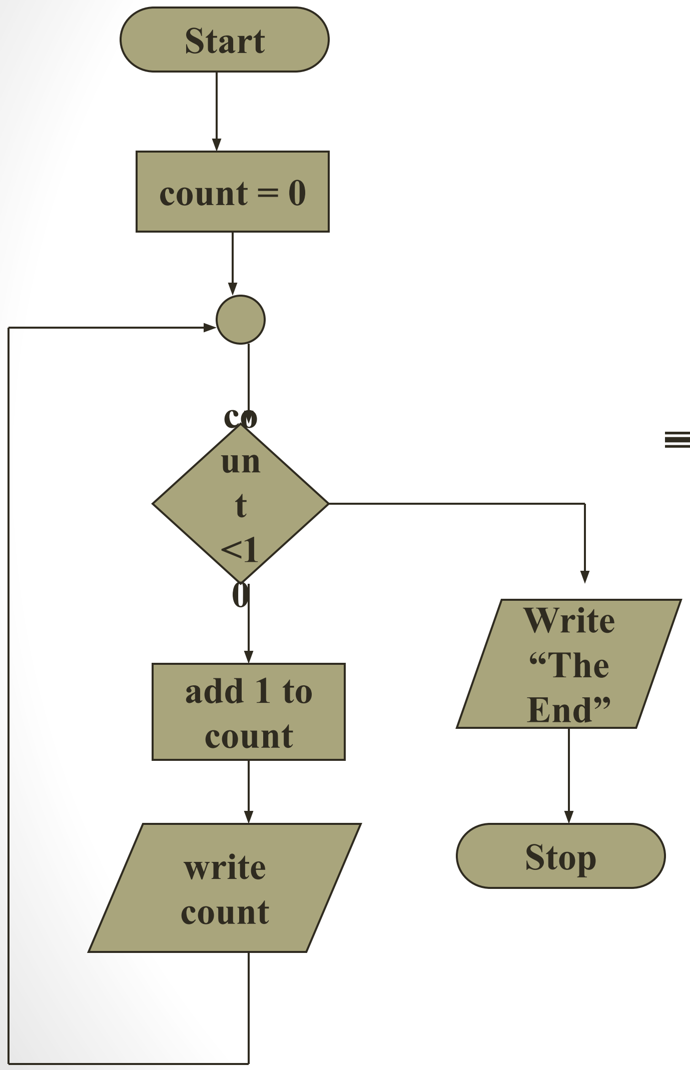
The Selection Structure



Pseudocode □

```
IF amount < 100
        interestRate = .06
ELSE
        Interest Rate = .10
ENDIF
```

WHILE / ENDWHILE



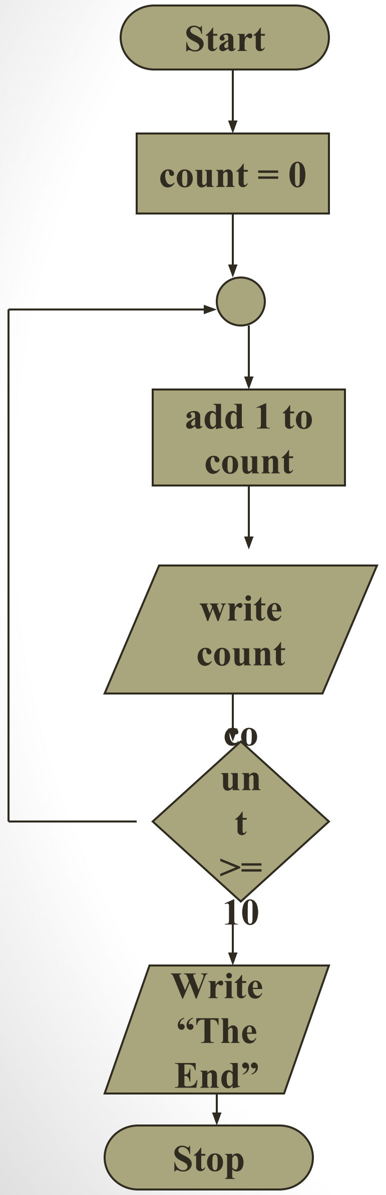
```
count = 0
WHILE count < 10
        ADD 1 to count
        WRITE count
ENDWHILE
WRITE "The End"
```

- Modular

```
Mainline
count = 0
WHILE count < 10
        DO Process
ENDWHILE
WRITE "The End"

Process
ADD 1 to count
WRITE count
```

REPEAT / UNTIL



```
count = 0
REPEAT
        ADD 1 to count
        WRITE count
UNTIL count >= 10
WRITE "The End"
```

- Modular

Mainline

```
count = 0
REPEAT
        DO Process
UNTIL count >= 10
WRITE "The End"
```

Process

```
ADD 1 to count
WRITE count
```

Advantages & Disadvantages

Flowchart Advantages:

- ✔ Standardized
- ✔ Visual

Pseudocode Advantages

- ✔ Easily modified
- ✔ Implements structured concepts
- ✔ Done easily on Word Processor

Flowchart Disadvantages:

- ✔ Hard to modify
- ✔ Structured design elements not implemented
- ✔ Special software required

Pseudocode Disadvantages:

- ✔ Not visual
- ✔ No accepted standard, varies from company to company

- What are the rules when writing pseudocode?
- What is the difference between pseudocode and a programming language?
- Write pseudocode for the following problem:

```
Find the average of 4 numbers and display it
```

```
input 4 numbers
sum=add numbers together
avg=sum/4
print avg
```

- https://en.wikibooks.org/wiki/GCSE_Computer_Science/Pseudocode
- https://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Problem_Solving/Pseudo_code

- https://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Problem_Solving/Pseudo_code
- https://en.wikibooks.org/wiki/GCSE_Computer_Science/Pseudocode