CHAPTER 8

Program Design & Algorithms

Information Technology for CSEC, 2nd Edition, Leo Cato…

(Pages 56 – 83)

OBJECTIVES
(Page 56)

1. Explain what is an algorithm
2. Identify variables and constants and distinguish between them
3. Use the correct data types to solve a problem
4. Be able to represent an algorithm in a variety of ways
5. Develop and write your own algorithm
6. Test algorithms

SIX STEPS TO SOLVE A PROBLEM

PROBLEM SOLVING PHASE

1. Identify & Define The Problem
2. Analyze The Problem & Break It Into Components
3. **Develop An Algorithm (Chapter 8)**
4. **Test The Algorithm To See If It Works (Chapter 8)**

IMPLEMENTATION PHASE

1. Write A Program In Programming Language
2. Test & Debug The Program

What Is An Algorithm?
(Page 56)

- An algorithm is a set of instructions to solve a given problem. Algorithms can be written as a **narrative**, or they can be written in **pseudocode**.

1. **Narrative** is writing out in the step-by-step solution to the problem in full sentences.
2. **Pseudocode** is language that mimics real programming language.

NARRATIVE & PSEUDOCODE
(Page 56)

Read in three decimal numbers. Find the total and the average. Display the average

Narrative Algorithm

Read in three numbers
Find the total
  Total = number1 + number2 + Number3
Find the average
  Average = total/3
Print the average

Pseudocode

Identify Constants
Numbers: an integer = 3
Identify Variables
Number1, Number2,Number3: a real number (three decimal numbers)
Total: a real number
Average: a real number
Processing
Read Number1, Number2, Number3
Total = number1 + number2 + number3
Average = total/numbers
Print Average
End Processing

IMPORTANT TOPICS

- **CATEGORIES OF DATA**

1. Constants
2. Variables

- **ELEMENTARY DATA TYPES**

1. Integers
2. Real Numbers
3. Character
4. String
5. Boolean

Naming Rules (Constants / Variables)

Constant Or Variable Name:

1. Cannot Start With A Number
2. Cannot Contain Any Spaces
3. Cannot Contain Symbols except underscore
4. Cannot be more than two words long
5. Cannot be called Constant1 or Variable1

What Is A Constant?
(Page 56)

- A **constant** is a data that is assigned a value and keeps that value throughout the program. It can be a known fact (scientific or mathematic)
- A constant does not change its value and does not depend on other factors. Example: value of pi = 3.1428 or your school = “San Pedro High School”
- Every constant has 3 parts: 1) constant name, 2) a data type, and 3) a value.

What Is A Variable?
(Page 56)

- A **variable** can change value in a program. Variables are: **user input (what is entered) or results of calculation in a program**.
- Every variable has 3 parts: 1) variable name, 2) a data type and 3) a description.

Elementary Data Types
(Page 57)

1. Integer
2. Real Number
3. Character
4. String
5. Boolean

Elementary Data Types
(Page 57)

- **INTEGER**: is a whole number and CANNOT have decimal points and cannot represent fractions. They can be represented as positive or negative numbers.
- For example: 5, 18, -20, 0, 5000

Elementary Data Types
(Page 57)

- **REAL NUMBER**: is a number that CAN include decimal points. They are sometimes called **floating point numbers** and they can be positive or negative numbers. Monetary (currency) values are always represented as real numbers.
- For example: 10.25, -5.75, 350.1, 20.00
- **Note: Only integers and real numbers are used in math formulas

Elementary Data Types
(Page 57)

- **CHARACTER**: is a SINGLE character including single letters of the alphabet or symbols.
- For example: M, F, $

Elementary Data Types
(Page 57)

- **STRING**: is a GROUP of characters. A string can be any number of characters.
- For example: San Pedro High, Belize, asdadad

Elementary Data Types
(Page 57)

- **BOOLEAN**: is a logical data type, having only two values (usually denoted true and false), intended to represent the truth values of logic.
- For Example: Yes or No

Identifying Variables, Constants, Data Types (Example 1)

- Look at the following problem and design an IPO Chart:
- Calculate the total salary earned after a month of weekly salaries. Show the results. (Note each week salary has a different amount)

Calculate the total wage earned after a month of weekly wages. Show the results. (Note each week salary is has a different amount)

INPUT	PROCESSING	OUTPUT
A month of weekly wages SAY week1 Week2 Week3 week4	1. Enter a month of weekly wages 2. Calculate the total wages MonthlySalary = week1 + week2 + Week3 + week4 3. Show the total wages	The total wages SAY monthlysalary

Variables:

Week1…Week4	Data Type: Real Number	Description: four weekly wages
MonthlySalary	Data Type: Real Number	Description: A month of salary

Identifying Variables, Constants, Data Types (Example 2)

- Look at the following problem and design an IPO Chart for it:
- Enter the daily temperature in Fahrenheit for three days. Find the average and then convert it to Celsius. Display both the results. (Conversion Formula: C = F – 32 * 0.5556

Enter the daily temperature in Fahrenheit for three days. Find the average and then convert it to Celsius. Display both the results. (Conversion Formula: C = F – 32 * 0.5556

INPUT	PROCESSING	OUTPUT
Three days of temperature in Fahrenheit SAY temp1 temp2 temp3	1. Enter three days of temperature in Fahrenheit 2. Find the average temperature average = (temp1 + temp2 + temp3)/3 3. Convert it to Celsius average_celsius = average – 32 * 0.5556 4. Display the average in Fahrenheit and Celsius	The average in Fahrenheit and Celsius SAY average Average_celsius

Variables:

temp1…temp3	Data Type: Real Number	Description: three Fahrenheit temperatures
Average	Data Type: Real Number	Description: The average Fahrenheit temperature
Average_celsius	Data Type: Real Number	Description: The average Celsius temperature

Constants:

Temperature	Data Type: Integer	Value: 3

ASSIGNING NAME RULES

- **Rules For Constant/Variable Names:**

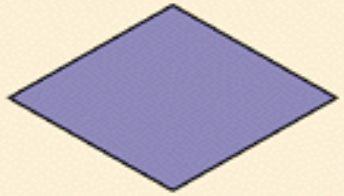
1. Cannot Contain Spaces
2. Cannot Start With A Number
3. Cannot Contain Any Symbols EXCEPT Underscore
4. Cannot Be Named: Constant1, Constant2, etc…
5. Cannot Be Named: Variable1, Variable2, etc…

FLOWCHARTS (Page 60)

- **Flowcharts** are geometrical diagrams that arrange the components of a problem (input, processing, output) in a logical sequence, which helps to avoid logic errors. The shapes are linked using arrowed lines that point to the next step in the sequence.

FLOWCHART SYMBOLS

ATTENTION!

Name	Symbol	Use in flowchart
Oval		Denotes the beginning or end of a program.
Flow line		Denotes the direction of logic flow in a program.
Parallelogram		Denotes either an input operation (e.g., INPUT) or an output operation (e.g, PRINT).
Rectangle		Denotes a process to be carried out (e.g., an addition).
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes (e.g., IF/THEN/ELSE).

FLOWCHART RULES

1. Must Use Ruler To Draw Symbols
2. Line Must Be Arrowed/Straight To Indicate Data Flow Direction
3. Flowchart Begins with word **START**
4. Flowchart Ends with **STOP**
5. For Input, Use only keyword **READ**
6. For Output, use only keyword **PRINT**
7. Declare Constants in a rectangle shape after **START**
8. Math Formulas Use (Addition +, Subtraction - , Multiplication *, Division /)
9. Use **BODMAS** (**B**racket **O**rder **D**ivision **M**ultiplication **A**ddition **S**ubtraction)

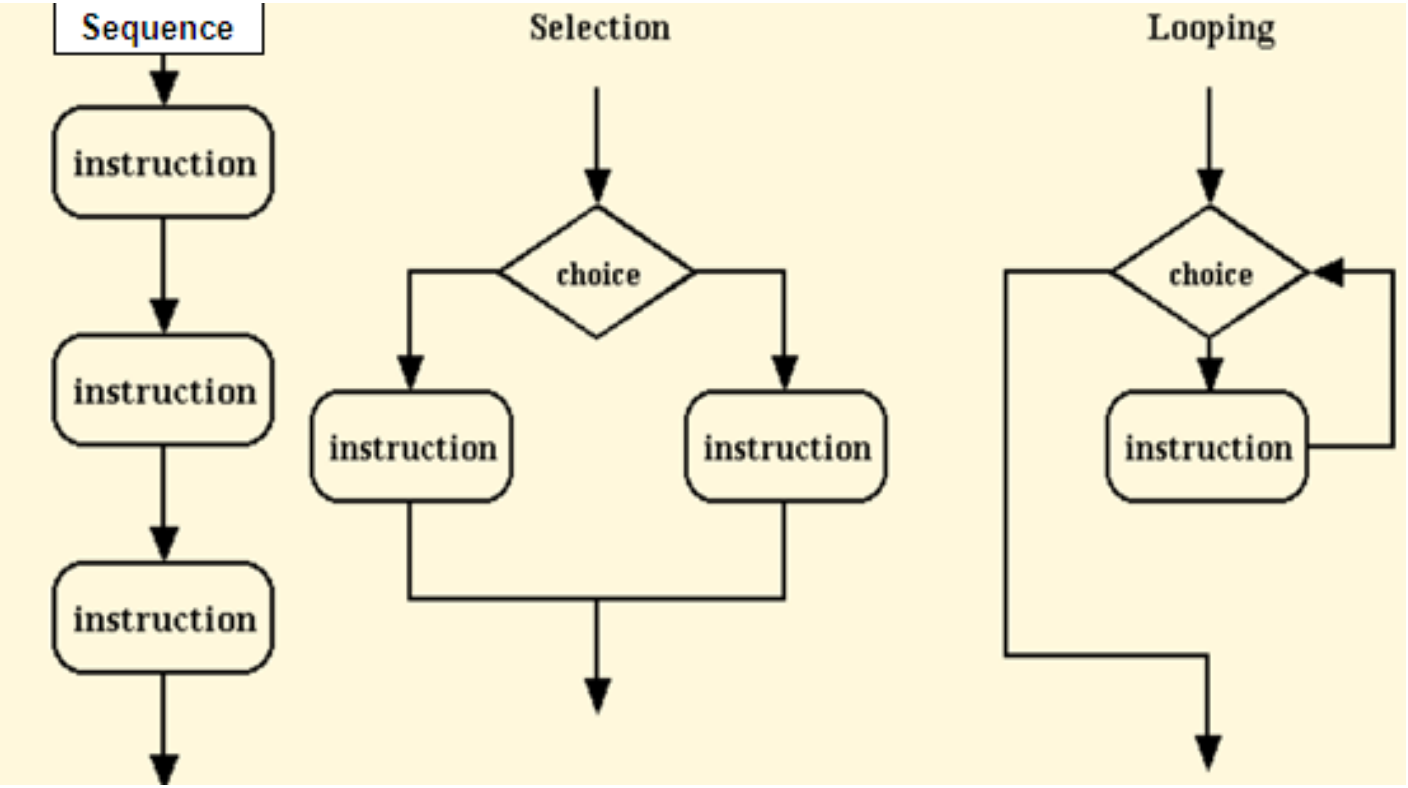
- Math Syntax: Solution = Math Equation
- Eg.: Total = price1 + price2 + price3

Types Of Flowcharting

- In flowcharting, the following terms are commonly used:

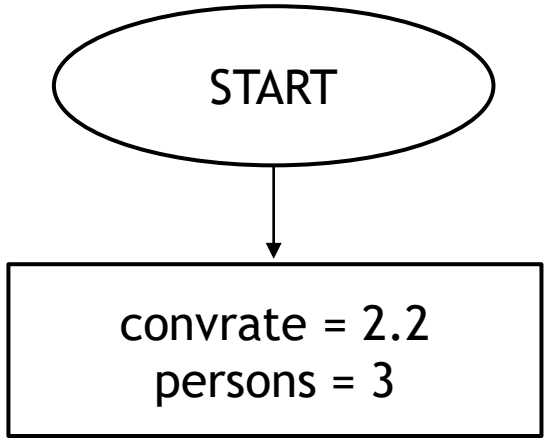
1. **Sequencing**: Execute one single instruction, in a section of program, after another.
2. **Selection**: Choose, depending on a tested condition, between two, or more pathways through a section of a program.
3. **Repetition/Looping**: Executing a single instruction, or group of instructions, one or more times.

Types Of Flowcharting



Representing Constants In Flowcharts

- Constants are represented in a RECTANGLE right after the START oval shape.
- Syntax Is: ***CONSTANTNAME = VALUE***
- Read in the weights of 3 person in kilograms. Calculate the average and display it in pounds. (1 kg = 2.2 lbs)



FLOWCHARTING – EXAMPLE #2

Enter five weights in pounds. Calculate the average and convert it to kilograms. Display both results(1lb = 0.453 kg)

FLOWCHARTING – EXAMPLE #2

IPO Chart

Input

Processing Enter five weights in pounds. Calculate the average and convert it to kilograms. Display both results(1lb = 0.453 kg) ***Output***

INPUT	PROCESSING	OUTPUT
Five weights in pounds Say Weight1, Weight2, Weight3, Weight4, Weight5	1. Enter Weight1, Weight2, Weight3, Weight4, Weight5 2. Calculate the average Avg_lbs = (Weight1 + Weight2 + Weight3 + Weight4 + Weight5)/5 3. Convert it to kilograms Avg_kg = Avg_lbs * 0.453 4. Display the average in pounds and kilograms	Average in pounds and kilograms Avg_lbs Avg_kg

FLOWCHARTING – EXAMPLE #2

Enter five weights in pounds. Calculate the average and convert it to kilograms. Display both results(1lb = 0.453 kg)

Constants
Name: Persons
Data Type: integer
Value: 5

Constants
Name: Conv_kg
Data Type: real number
Value: 0.453

Variables
Name: weight1…Weight5
Data Type: real number

Variables
Name: Avg_lbs, Avg_kg
Data Type: real number

This remains the same thru the entire program

User Input is normally a variable

Result of a Calculation is normally a variable

FLOWCHARTING - #2

Constants are declared in a rectangle symbol.
Use the constant name equal to a value

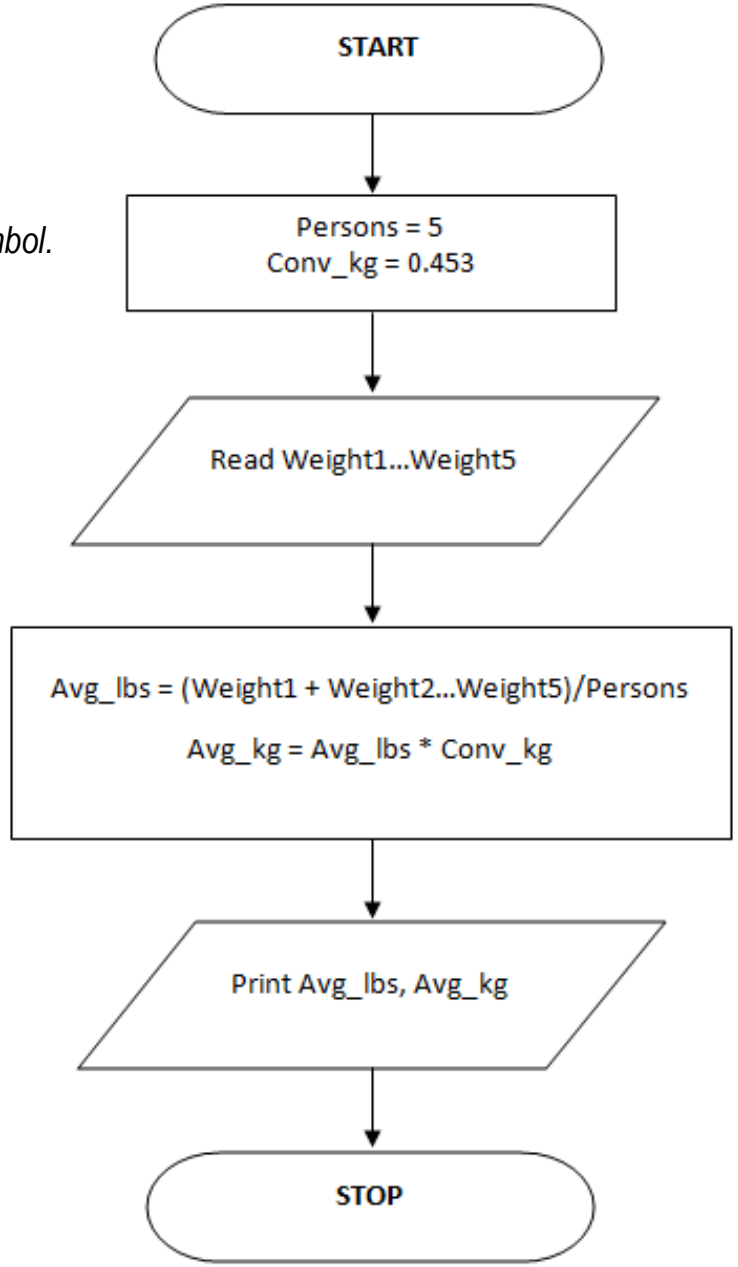
Check this out!

Use the keyword READ and variable name. If multiple variable names then separate by commas

check this out

Create the math formula using the variable & constant names

Use the keyword PRINT and variable name to show the solution



PSEUDOCODE

Page 62

Pseudocode

- **Pseudocode** is similar to actual programming language, but it does not use exactly the same terminology. Pseudocode helps to familiarize you with the syntax and structure of a program.

Pseudocode

- Pseudocode follows the following general coding syntax:
- Only one statement should occur per line.
- Indentation of three spaces within conditional and loop statements. Programmers indent to better convey the structure of their program to human readers.
- **SPELLING ERRORS ARE NOT ACCEPTABLE.**

Pseudocode Template

Identify variables

Variable declarations

Identify constants

Constants declarations

Processing

Input components

Processing components

Output components

End Processing

TEST TIP:
Be able to identify the main sections of a pseudocode.

Identify Variables

- Follow these steps to structure the pseudocode:

1. Create a section at the start of the program that identifies variables. (IDENTIFY VARIABLES)
2. Declare the data type for each.
3. Explain what each of those values is in brackets.

- **Syntax:**

Name: a/an data type (description)

Identify Variables

- Example:

Enter the price of three items. Compute the subtotal and then add in a GST tax of 12.5% to get a final total. Display the final result.

IDENTIFY VARIABLES

Price1, Price2, Price3: a real number (the prices of the items)

Subtotal: a real number (the subtotal after adding items)

Tax: a real number (the tax on the subtotal)

Total: a real number (the final total)

Identify Constants

- Follow these steps to structure the pseudocode:

1. Create a second section that identifies constants. (IDENTIFY CONSTANTS)
2. Declare the data type.
3. Declare the value for each.

- **Syntax:**

  Name: a/an data type = value

Identify Variables/Constants

- Example:

Enter the price of three items. Compute the subtotal and then add in a GST tax of 12.5% to get a final total. Display the final result.

IDENTIFY VARIABLES

Price1, Price2, Price3: a real number (the prices of the items)

Subtotal: a real number (the subtotal after adding items)

Tax: a real number (the tax on the subtotal)

Total: a real number (the final total)

IDENTIFY CONSTANTS

Price: an integer = 3

gsttax: a real number = 0.125

Identify Variables/Constants

- Example:

Enter the age of a student, halve it and display the result.

IDENTIFY VARIABLES

Age: an integer (the age of a student)

New_Age: a real number (the new age divided by 2)

IDENTIFY CONSTANTS

Half: an integer = 2

Processing

- Follow these steps to structure the pseudocode:

1. Create a third section that allows IPO statements. (PROCESSING)
2. All statements will go in here: Input, Processing and Output

Processing – Input Component

- **Input**
- In pseudocode and actual code, you use **commands**. These are words that tell the computer to perform a specific action. For input statements, the pseudocode commands are **READ and INPUT**.
- (Recommendation: Use only READ since it is also the keyword used in flowcharting)
- **Syntax:**

  Read *variablename(s)*

Processing (Example)

Enter the price of three items. Compute the subtotal and then add in a GST tax of 12.5% to get a final total. Display the final result.

IDENTIFY VARIABLES

Price1, Price2, Price3: a real number (the prices of the items)

Subtotal: a real number (the subtotal after adding items)

Tax: a real number (the tax on the subtotal)

Total: a real number (the final total)

IDENTIFY CONSTANTS

items: an integer = 3

Gst: a real number = 0.125

PROCESSING

Read Price1, Price2, Price3

Processing – Output Component

- **Output**
- The output statements will tell you what result is shown. Two output commands in pseudocode are **PRINT and OUTPUT**
- (Recommendation: Use only PRINT since it is also the keyword used in flowcharting)
- **Syntax:**

  Print *variablename(s)*

Processing – Output Component

It is very important to understand the difference between printing out a value that has been calculated and printing out an actual word or statement. Here the computer will print the value that has been calculated for TotalPrice, not the phrase "TotalPrice". However, you can and may want to print a message "The total price is$" to accompany the value. If you write the below statement to do this:

Print The total price is $ TotalPrice 

This is a WRONG format and the computer will not be able to carry this out. By default, characters or strings that follow a PRINT command are assumed as identifiers that have already been declared. In the statement above, the only known identifier is TotalPrice

To print information that has not been stored and calculated in a program, you enclose it in quotation marks so the computer does not try to interpret that data:

Print "The total price is $",TotalPrice 

This is a CORRECT format and the string enclosed in quotation marks will be printed first, followed by whatever value has been calculated for TotalPrice. A comma separates the items so they cannot be interpreted as one item. If you run this statement on a computer, this is the output you will see on the screen:

The total price is $34.50

Processing (Example)

Enter the price of three items. Compute the subtotal and then add in a GST tax of 12.5% to get a final total. Display the final result.

IDENTIFY VARIABLES

Price1, Price2, Price3: a real number (the prices of the items)

Subtotal: a real number (the subtotal after adding items)

Tax: a real number (the tax on the subtotal)

Total: a real number (the final total)

IDENTIFY CONSTANTS

Gst: a real number = 0.125

PROCESSING

Print “Enter the price of the three items”

Read Price1, Price2, Price3

Print “The final total is $“,Total

END PROCESSING

User Prompt Message Makes A Program User Friendly

The Math Formulas will go in the blank space here.

Processing - Calculation Tables

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division

Processing – Relational Operators

Operator	Meaning
<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To
<>	Not Equal To
=	Equal To

Processing - Conditional/Looping Operators

Operator	Purpose
IF	Compares a statement against a condition to see if it is true or false
THEN	Executes an instruction when a condition is true
AND	Links two or more conditions that have to be met
OR	Provides an extra condition
ELSE	Executes an instruction should a condition be false
FOR	Creates a loop that is carried out a known number of times
WHILE	Creates a loop that is carried out an unknown number of times

Processing – Math Equations

- In pseudocode, math equations are created similar to equations done in flowchart.
- The same formula designed in the flowchart step is **THE SAME (LO MISMO)** formula to be used in the pseudocode.
- **Syntax:**

  Solution = Math Equation

Complete Pseudocode (Example #1)

Enter the price of three items. Compute the subtotal and then add in a GST tax of 12.5% to get a final total. Display the final result.

IDENTIFY VARIABLES

Price1, Price2, Price3: a real number (the prices of the items)

Subtotal: a real number (the subtotal after adding items)

Tax: a real number (the tax on the subtotal)

Total: a real number (the final total)

IDENTIFY CONSTANTS

GST: a real number = 0.125

PROCESSING

Read Price1, Price2, Price3

Subtotal = Price1 + Price2 + Price3

Tax = Subtotal * GST

Total = Subtotal + Tax

Print “The final total is $“,Total

END PROCESSING

The same Math Formulas created in the flowchart.

Complete Pseudocode (Example #2)

Enter the age of a student, halve it and display the result.

IDENTIFY VARIABLES

Age: an integer (the age of a student)

New_Age: a real number (the new age of a student)

IDENTIFY CONSTANTS

Half: an integer = 2

PROCESSING

Read Age

New_Age = Age / Half

Print "The new age of the student is ",New_Age

END PROCESSING

The same Math Formula created in the flowchart.

Complete Pseudocode (Example #3)

Enter the heights of the starting 5 players on a team. Compute the average height in feet and then convert it to meters. Display the final average in meters. (1 Foot = 0.3048 meter)

IDENTIFY VARIABLES

Height1…Height5: a real number (the height of the players in feet)

Average_Ft: a real number (the average height in feet)

Average_Meter: a real number (the average in meters)

IDENTIFY CONSTANTS

Players: an integer = 5

Convrate: a real number = 0.3048

PROCESSING

Read Height1…Height5

Average_Ft = (Height1+Height2…Height5)/players

Average_Meter = Average_Ft * convrate

Print "The final average in meters is ",Average_Meter

END PROCESSING

The same Math Formulas used in the flowchart.

Complete Pseudocode (Example #4)

Enter the number of daily absents for the class of 1A for only the Fridays in the month of September 2015. Calculate the total and average absents for the days requested and display both the results.

IDENTIFY VARIABLES

Absents1…Absents4: an integer (the Friday absents for the month)

Total: an integer (the total absents for the Fridays)

Average: a real number (the average absents for the Fridays)

IDENTIFY CONSTANTS

Fridays: an integer = 4

Form: a string = “1A”

Month: a string = “September 2015”

PROCESSING

Read Absents1…Absents4

Total= Absents1+Absents2…Absents4

Average = Total / Fridays

The same Math Formulas used in the flowchart.

Print “The total absents for class “,Form,” for the month of “,Month,” is “,Total,” and the average is “,average

END PROCESSING

Complete Pseudocode (Example #5)

Enter the number of tourists that disembark from the cruise ships into the Tourism Village on a day. Each tourist pays a entrance fee of $10.00 USD and a visitor tax of $3.00 USD. Calculate how much the Tourism Village would collect on a day in both USD and BZD. Display both results

IDENTIFY VARIABLES

Tourist: an integer (the number of tourists from the cruise ship)

TotalUSD: a real number (the total collection in USD)

TotalBZD: a real number (the total collection in BZD)

IDENTIFY CONSTANTS

Entrance_Fee: a real number = 10.00

Visitor_Tax: a real number = 3.00

Conversion: a real number = 2.00

PROCESSING

Read Tourist

TotalUSD= (Tourist * Entrance_Fee) + (Tourist * Visitor_Tax)

TotalBZD = TotalUSD * Conversion

Print “The total collection in US Dollars is $“,TotalUSD,” and in BZ Dollars is $“,TotalBZD

END PROCESSING

The same Math Formulas used in the flowchart.

Conditional Statements (Pg. 66)

- A problem may have options that lead to different solutions where values are compared against a condition, which is a set of criteria. If the criteria are met, the condition is said to be true. The path taken depends on whether the condition is true or false, this is the task of a conditional statement.

- **Syntax:**

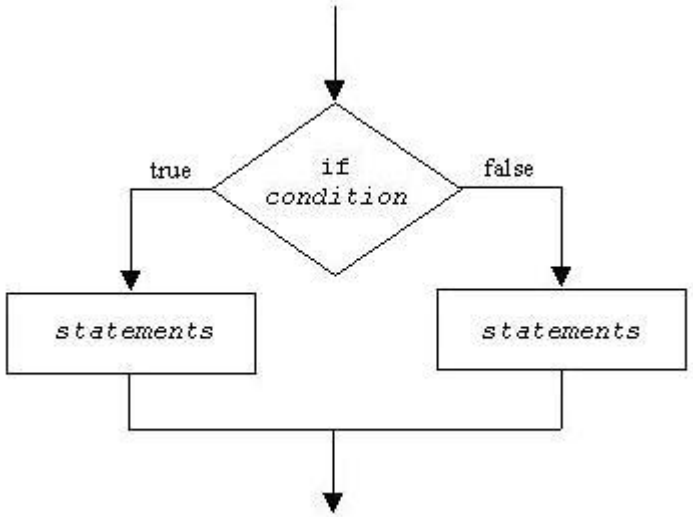
  IF logical test THEN

  true path

  ELSE

  false path

  ENDIF



Complete Pseudocode (Conditional)

Enter the name and grade of a student. If the grade is greater than or equal to 70, Display a “Pass” message along with the student name and grade. If it is less than 70, Display a “Fail” message along with the student name and grade.

IDENTIFY VARIABLES

grade: a real number (the grade of a student)

student: a string (the name of a student)

IDENTIFY CONSTANTS

passmark: an integer = 70

PROCESSING

Read student, grade

IF grade >= passmark THEN

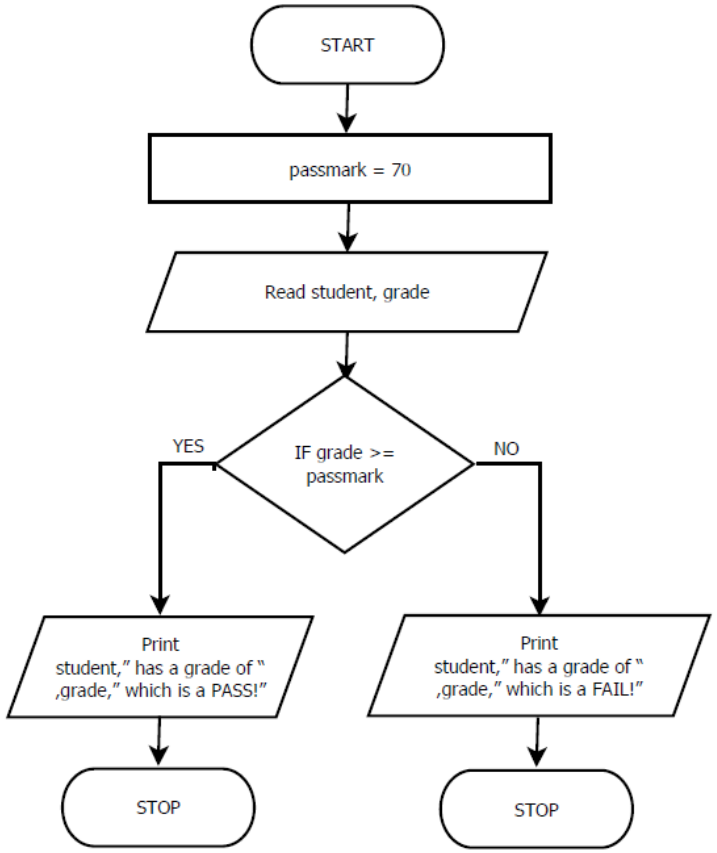
  Print student,” has a grade of “,grade,” which is a PASS!”

ELSE

  Print student,” has a grade of “,grade,” which is a FAIL!”

ENDIF

END PROCESSING



Complete Pseudocode (Conditional)

Calculate the Xmas discount given to a customer based on the value of their purchases. Add the prices of a customer's items (a maximum of 3 items) to get a total. If the total is under $50.00, the customer gets no discount. If the total is equal to or greater than $50.00, the customer gets a 15% discount. Show the amount of the discount and the discounted total price on the receipt.

PROCESSING

Read price1, price2, price3

Total = price1 + price2 + price3

IF total >= disc_mark THEN

  discount = total * rate

ELSE

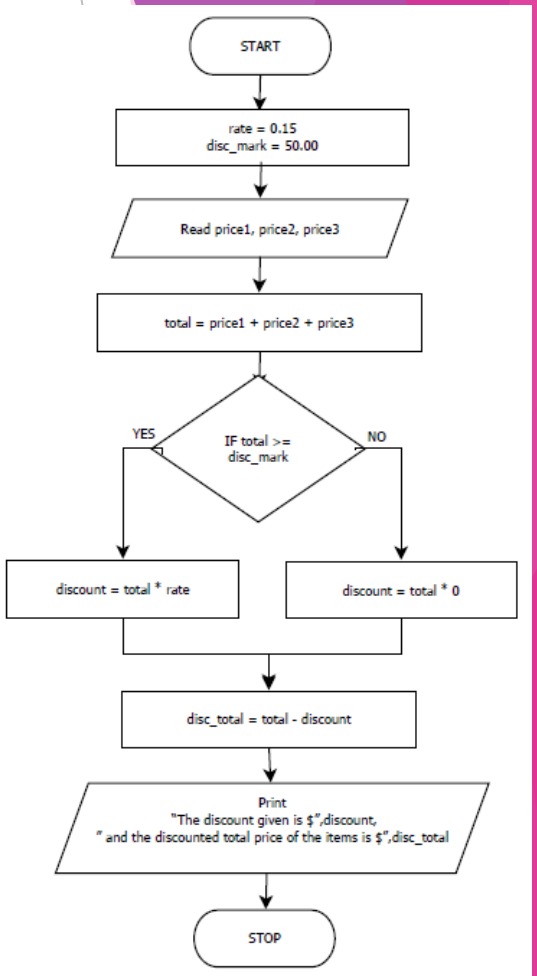
  discount = total * 0

ENDIF

disc_total = total – discount

Print "The discount given is $",discount," and the discounted total price of the items is $",disc_total

END PROCESSING



Nested IF (Pg. 67)

- It is possible to have more than two options available in a problem. If this is the case, the construct becomes IF-THEN-ELSE-IF, and so on. You can have as many sets of IF-THEN-IF within a problem as you need. This is referred to as **NESTED IF** statements.
- Sometimes you may want a value to be measured against more than one condition at a time. This is when you use the **AND** and **OR** operators. They are also called **Boolean operators**. Consider this problem:
- **Syntax:**

  IF logical test with AND/OR THEN

  true path 1

  ELSE

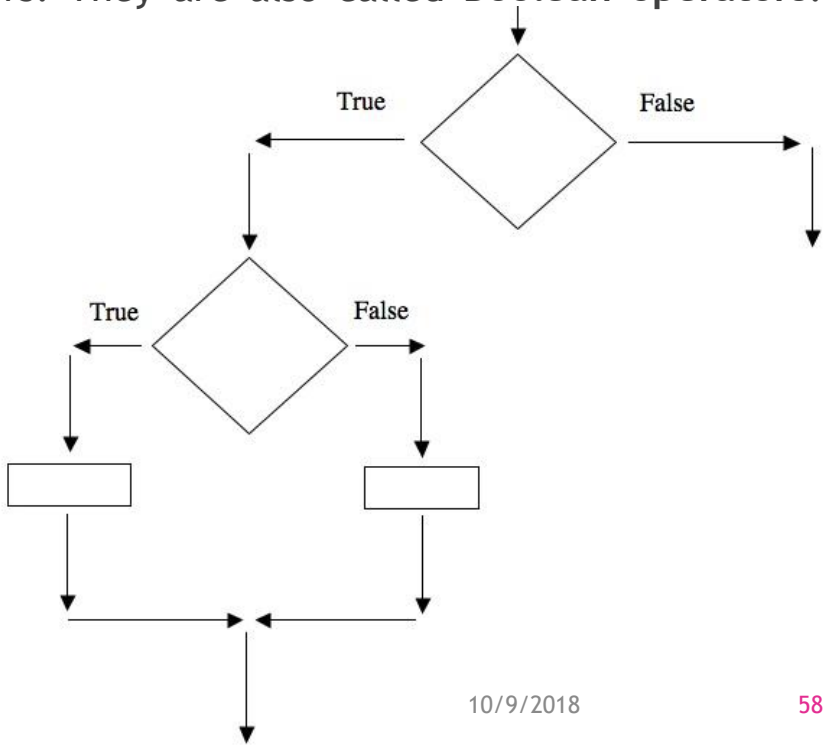
  IF logical test with AND/OR THEN

  true path 2

  ELSE

  false path

  ENDIF



Complete Pseudocode (Nested IF)

Input the percentage received in a test. If the percentage is greater than or equal to 70 and less than or equal to 80, display “Satisfactory”. If it is greater than 80, display “Good”. If it is less than 70, display “Fail”

PROCESSING

Print “ Enter the grade of a student”

Read grade

IF grade >=passmark **AND** grade <=goodmark **THEN**

  Print “Satisfactory”

ELSE

 IF grade > goodmark **THEN**

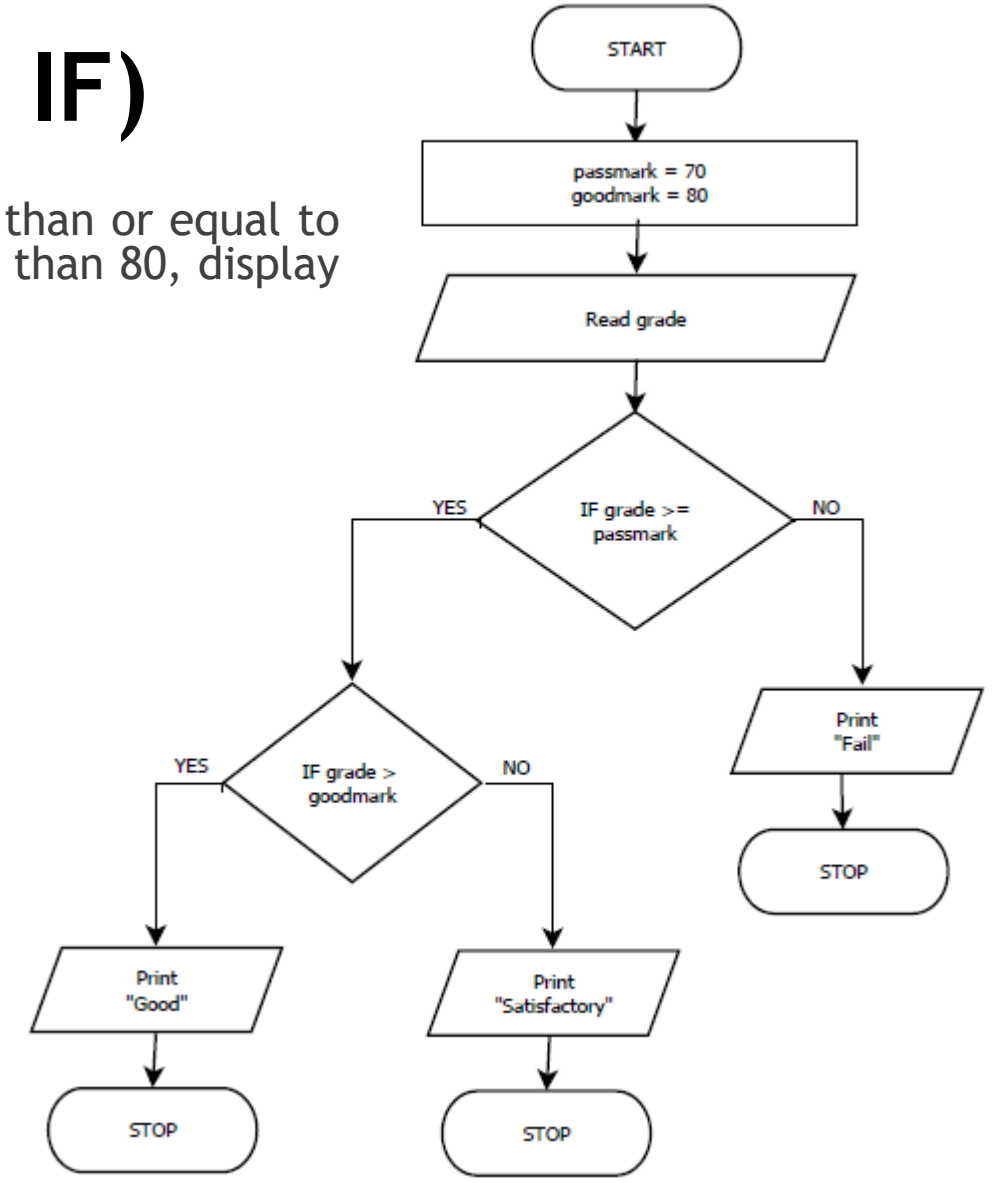
  Print “Good”

ELSE

  Print “Fail”

ENDIF

END PROCESSING



Complete Pseudocode (Nested IF)

Input a number between 1 and 5. If the number entered is 4 or 5, display "HIGH". If it is 1 or 2, display "LOW". If it is 3, display "MIDDLE".

IDENTIFY VARIABLES

IDENTIFY CONSTANTS

PROCESSING

Read number

IF number = 4 **OR** number = 5 **THEN**

  Print "HIGH"

ELSE

 IF number = 3 **THEN**

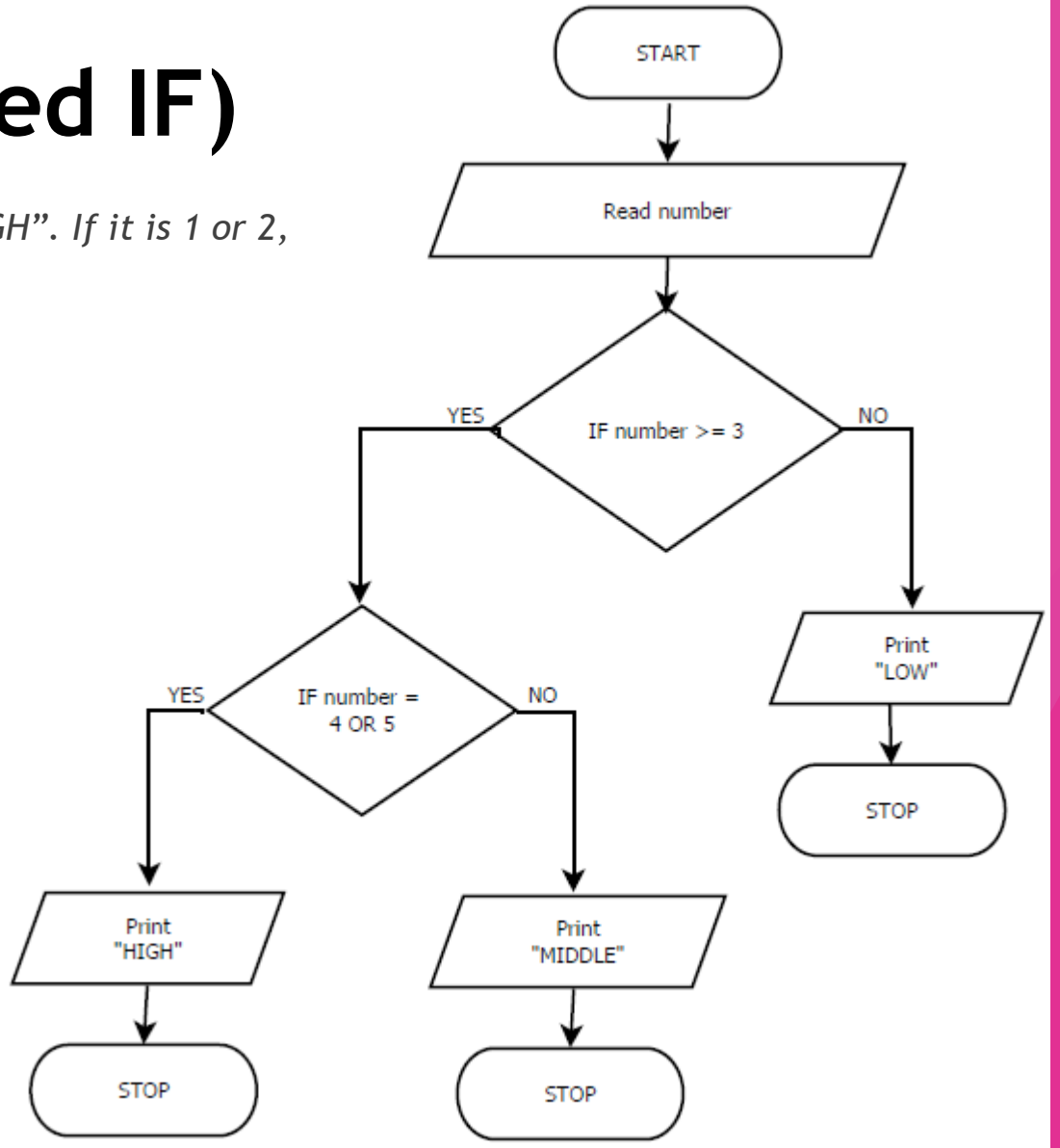
  Print "MIDDLE"

ELSE

  Print "LOW"

ENDIF

END PROCESSING



ASSIGNMENT (12/01/2017)

- **EXERCISE 8.5** on PAGE 70 (FOLDER SHEETS)
- QUESTIONS 1 to 3 – Complete Pseudocode
- Question 1 – Complete Pseudocode + Flowchart
- ***Note**: Students That Received A Zero Due On Last Assignment To Copying Will Do A Different Assignment In The ITLAB After Classes.*

TEST (12/06/2017 &12/07/2017) (1.5 Hour)

- Practical Proficiency Only
- Using TEST ACCOUNT and With Microsoft Word
- You Will Be Assessed On Pseudocode + Flowchart
- .Basic, Conditional (IF, Nested IF), Loops

Truth Tables

- Truth tables are very useful when using Boolean operators in conditional statements. A **truth table** lists all the possible values that can be achieved when you enter any possible combination of values into the problem.
- There are two types of truth tables:

1. AND Truth Table
2. OR Truth Table

AND Truth Tables

- The **AND** truth table is used when both conditions in a conditional statement must be true. The first column measures the first condition – *for the example problem, whether the* percentage is 70 or over. The second column measures the second condition – whether the percentage is 80 or less.
- The third column is the **OUTPUT** column. It shows the results when the first and second column conditions are met or not. **In the truth table rows, always use 1 to indicate a true statement, and 0 to indicate a false statement.**

Condition >=70	Condition <=80	Output Print "Satisfactory"
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Tables

- The OR truth table is used when there is a comparison against two or more values and only one condition has to be true. **In the truth table rows, always use 1 to indicate a true statement, and 0 to indicate a false statement.**

The truth table for the first part is:

Condition Number = 4	Condition Number = 5	Output Print "High"
0	0	0
0	1	1
1	0	1

LOOPS (Pg. 70)

- **LOOPS** are statements or instructions that get repeated a known amount of times or until the user decides to terminates the loop.
- There are three (3) loops

1. WHILE
2. REPEAT-UNTIL
3. FOR

- **INITIALIZE VARIABLES** are used for 2 purposes: 1) to assign a value to a variable for a particular problem; 2) to reset the variables to a zero value

WHILE LOOP (Pg. 70)

- A WHILE loop is used when you want to do a loop an indefinite number of times. To construct a WHILE statement, you assign an initial value to a variable and the instructions are repeated until that variable reaches a certain value or point.

Syntax:

WHILE logical test DO

Sequence

ENDWHILE

WHILE LOOP (Example #6)

- EXERCISE 6
- Design a user friendly algorithm that will input the daily ticket sales of a local cinema and keep an accrued total sales & the number times the loop is run until the algorithm is terminated when the user enters -1 for the daily ticket sales. The program should display the accrued total sales and the number of times the loop is executed. (WHILE)

WHILE LOOP (Example #2)

IDENTIFY VARIABLES

ticketsales:

totalsales:

counter: an integer (the number of times the loop is executed

WHILE LOOP (Pg. 70)

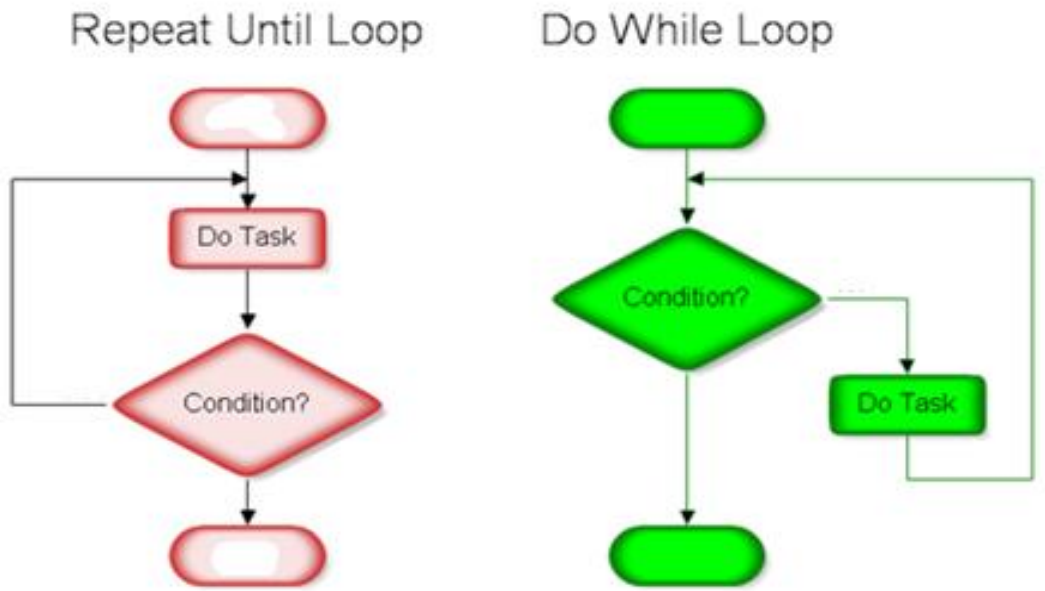
- There are five important aspects to note:

1) Initialization of variables should appear first in the processing section.
2) Any variable that is affected by or that depends on the calculation in the loop must be initialized.
3) The variable Signal is initialized to Y to make sure the loop starts, because the loop will not run if the value is not Y.
4) The variable Average is initialized because its value depends on a calculation within the loop.
5) The three grades do not have to be initialized because they depend on user input and do not change their value during the calculation.

WHILE LOOP (Exercise) (In Notebook)

- Amanda is a fitness consultant at BUFF N' TUFF gym and charges $20.00 per hour for clients. Write a pseudocode that will allow her to enter the number of hours she worked for the day and then calculate how much she has made daily. Amanda will determine when the loops stop with YES to continue or NO to stop.

LOOPS
WHILE vs. REPEAT FLOWCHART



REPEAT - UNTIL LOOP (Pg. 72)

- The **REPEAT** loop is similar to the WHILE loop. It performs a calculation an undetermined number of times by comparing a value against a condition until the condition is no longer true.

Syntax:

REPEAT

Sequence

UNTIL logical test

REPEAT-UNTIL LOOP (Example #1)

A number has a starting value of 100. Repeat a calculation of adding 5 to the number until it reaches 120 and display the answer after each calculation.

IDENTIFY VARIABLES

startnum

IDENTIFY CONSTANTS

increase

limit

What Have You Learnt?

PROCESSING

startnum = 100

REPEAT

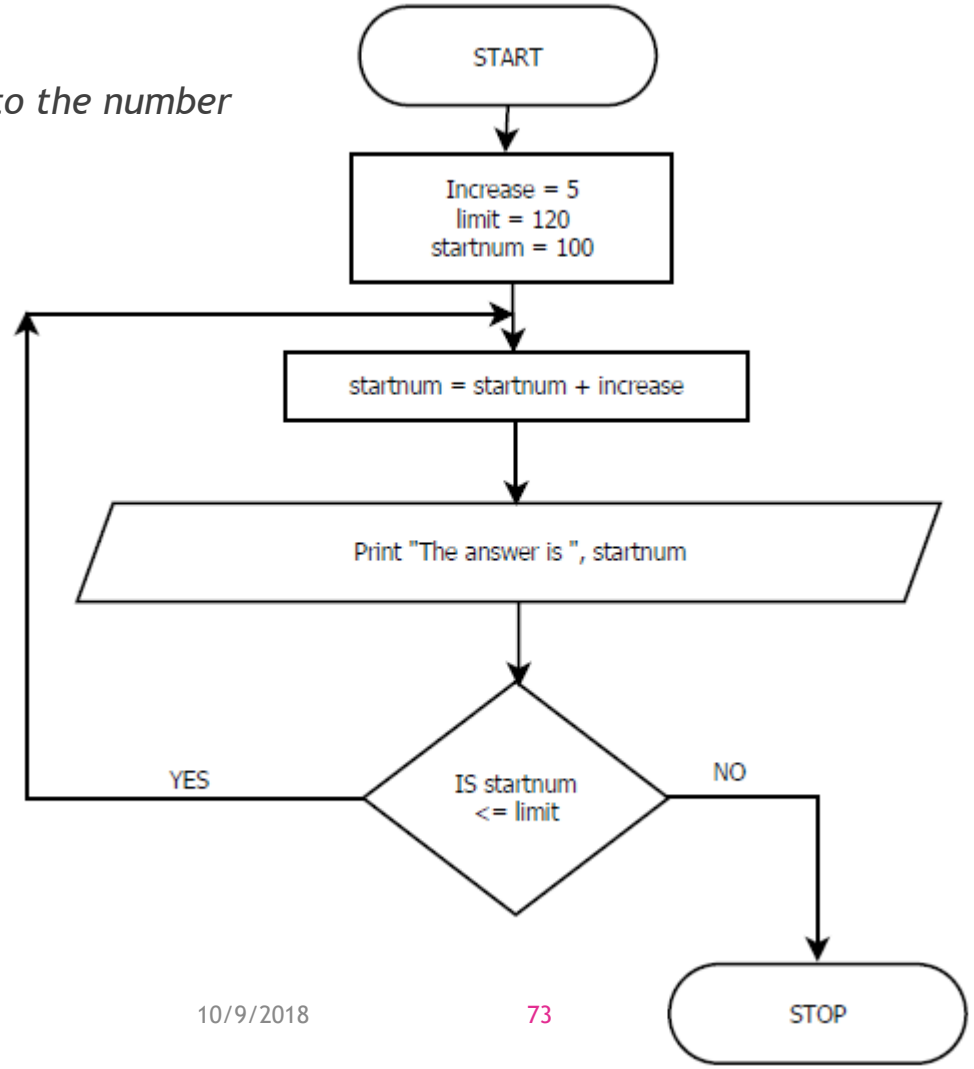
  startnum = startnum + increase

  Print “The answer is “, startnum

UNTIL startnum <= limit

Initializing of variable

END PROCESSING



FOR LOOP (Pg. 73)

- The FOR loop also relies on a condition, but unlike a WHILE loop, you set the start and end values for the variable, because you know how many times you want the loop repeated.

Syntax:

```
FOR counter = startvalue TO endvalue DO
    Sequence
ENDFOR
```

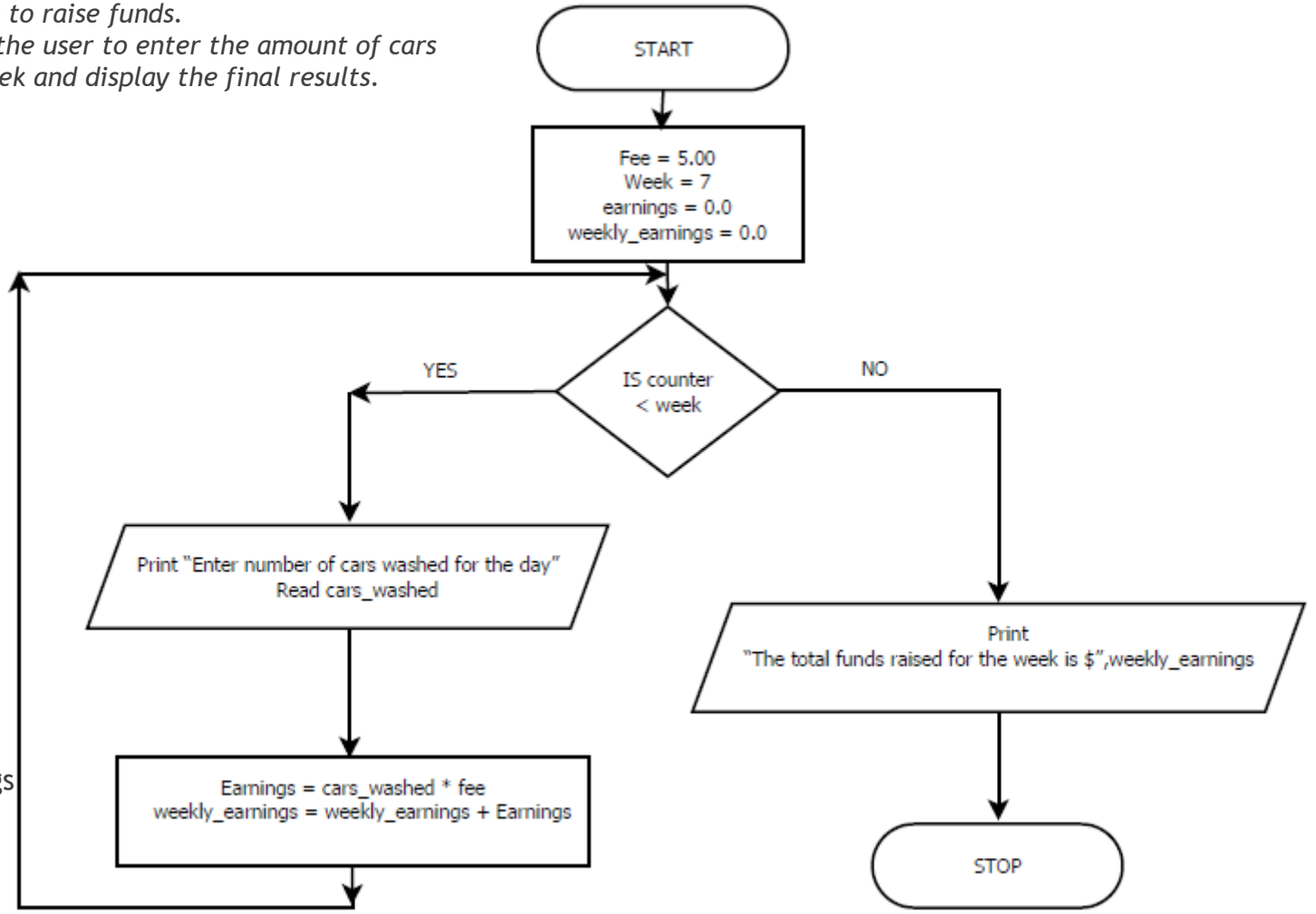
FOR LOOP (Example #1)

The San Pedro Cancer Society is doing a car wash for an entire week to raise funds.
They charge $5.00 per car wash, write an algorithm that will allow the user to enter the amount of cars washed daily and then calculate their earnings at the end of the week and display the final results.

IDENTIFY VARIABLES
cars_washed
Earnings
weekly_earnings
Counter

IDENTIFY CONSTANTS
Fee
Week

PROCESSING
Earnings = 0.0
weekly_earnings = 0.0
Initializing of variables
FOR counter = 1 **TO** week **DO**
  Print "Enter number of cars washed for the day"
  Read cars_washed
  Earnings = cars_washed * fee
  weekly_earnings = weekly_earnings + Earnings
ENDFOR
Print "The total funds raised for the week is $",weekly_earnings
END PROCESSING



FOR LOOP - Tables (Pg. 74)

- The FOR loop is useful for creating tables. You can create an entire set of values and print column names so it looks like a table. This portion of pseudocode prints out a conversion table for the 1 to 5 feet to yards/inches:

```
PROCESSING
yards = 0.0
Inches = 0
Print "FEET","YARDS","INCHES"
FOR counter = 1 to 5 DO
   yards = counter / convert_yards
   inches = counter * convert_inches
   Print counter, yards, inches
ENDFOR
END PROCESSING
```

FEET	YARDS	INCHES
1	0.33	12
2	0.67	24
3	1.00	36
4	1.33	48
5	1.67	60

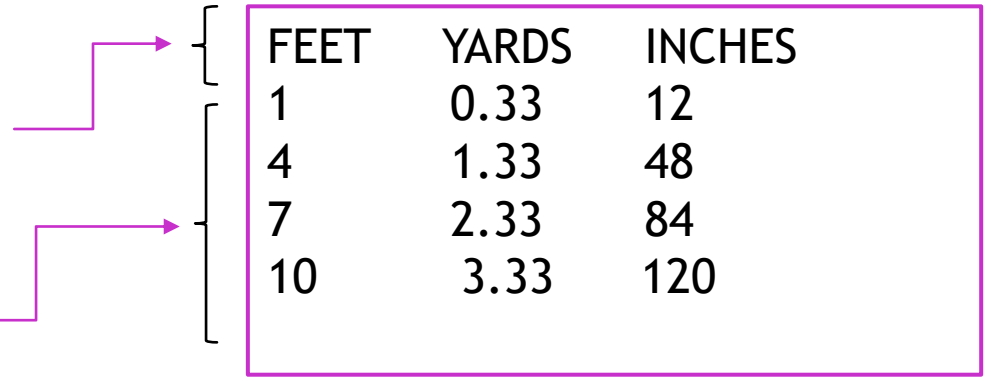
FOR LOOP - STEP Clause (Pg. 75)

- The FOR loop is useful If you want increments of 2, 3, etc., use the STEP clause. The STEP clause follows directly after the FOR statement. This portion of pseudocode prints out a conversion table for the 1 to 12 feet to yards/inches incrementing by 3:

PROCESSING

yards = 0.0
Inches = 0
Print “FEET”,”YARDS”,”INCHES”
FOR counter = 1 **to** 12 **STEP** 3 **DO**
  yards = counter / 3
  inches = counter * 12
  Print counter, yards, inches
ENDFOR

END PROCESSING



FEET	YARDS	INCHES
1	0.33	12
4	1.33	48
7	2.33	84
10	3.33	120

FOR LOOP – DOWN TO/ STEP (Pg. 75)

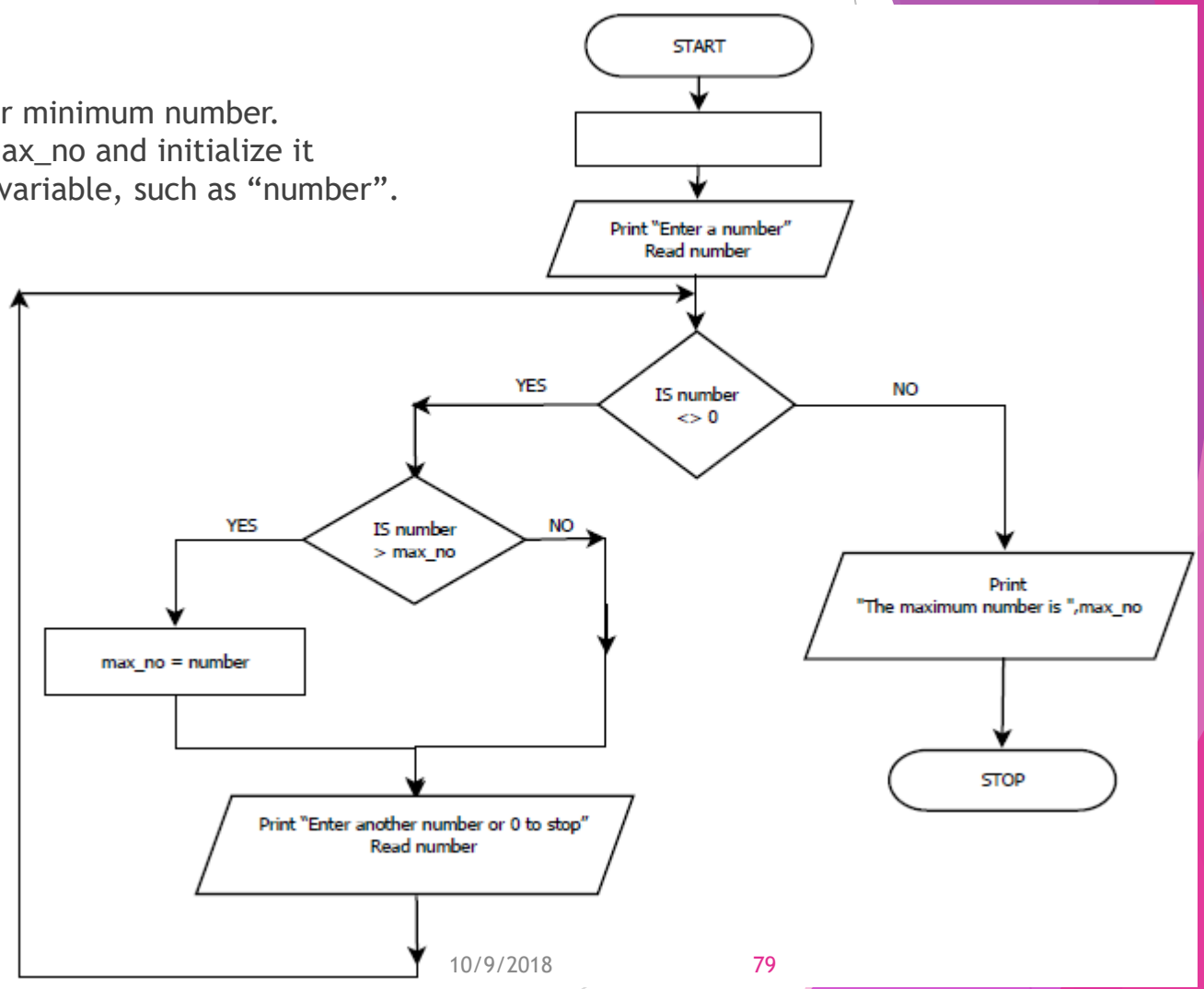
- If you are using a FOR loop that repeats in descending order and you want to use STEP, you put a negative sign in front of the step value and use it with the DOWN TO feature. The computer assumes that any value without a sign in front of it is a positive value. You must indicate a negative value when you want a negative value. This portion of a pseudocode displays every second number from 20 to 2:

```
FOR counter = 20 DOWN TO 2 STEP -2 DO
   Print counter
ENDFOR
```

WHILE LOOP - MAXIMUM (Pg. 75)

A FOR loop or WHILE loop can be used to find a maximum or minimum number.
To find a maximum number you create a variable such as max_no and initialize it with a low number. You then compare max_no to an input variable, such as “number”.
Here is part of the pseudocode:

```
IDENTIFY VARIABLES
number
max_no
PROCESSING
max_no = 0
Print “Enter a number”
Read number
WHILE number <> 0 DO
   IF number > max_no THEN
      max_no = number
   ENDIF
   Print “Enter another number or 0 to stop”
   Read number
ENDWHILE
Print “The maximum number is”, max_no
END PROCESSING
```



FOR LOOP - MAXIMUM (Pg. 75)

- A FOR loop or WHILE loop can be used to find a maximum or minimum number. To find a maximum number you create a variable such as max_no and initialize it with a low number. You then compare max_no to an input variable, such as “number”. Here is part of the pseudocode:

```
IDENTIFY VARIABLES
number
max_no
PROCESSING
max_no = 0
Read number
WHILE number <> 0 DO
   IF number > max_no THEN
      max_no = number
   ENDIF
   Print “Enter another number or 0 to stop”
   Read number
ENDWHILE
Print “The maximum number is”, max_no
END PROCESSING
```

FOR LOOP - Maximum (Example)

Enter the heights of 20 athletes in meters, converts the value to centimeters and displays the height of the tallest person in centimeters.

IDENTIFY VARIABLES
Height_m
Height_cm
highest
counter

IDENTIFY CONSTANTS
centimeters
athletes

PROCESSING
highest = 0
height_cm = 0

Initializing of variables

```
FOR counter = 1 TO athletes DO
   Print "Enter the height of the athlete in meters"
   Read height_m
   height_cm = height_m * centimeters
   IF height_cm > highest THEN
      highest = height_cm
   ENDIF
ENDFOR
Print "The height of the tallest athlete is ",highest," centimeters"
```

END PROCESSING

WHILE LOOP - MINIMUM (Pg. 76)

- A FOR loop or WHILE loop can be used to find a minimum number. To find a minimum number you create a variable such as min_no and initialize it with a high number. You then compare min_no to an input variable, such as “number”. Here is part of the pseudocode:

```
IDENTIFY VARIABLES
number
min_no
PROCESSING
min_no = 9999
Read number
WHILE number <> 0 DO
   IF number < min_no THEN
      min_no = number
   ENDIF
   Print “Enter another number or 0 to stop”
   Read number
ENDWHILE
Print “The smallest number is”, min_no
END PROCESSING
```

FOR LOOP - Minimum (Example)

Enter the heights of 11 football players on FC Barcelona in feet, convert the value to meters and displays the height of the shortest players in meters.

IDENTIFY VARIABLES
Height_feet
Height_meters
min_no
counter

IDENTIFY CONSTANTS
meters
players

PROCESSING
min_no = 9999
height_meters = 0.0

Initializing of variables

```
FOR counter = 1 TO players DO
   Print "Enter the height of a Barcelona player in feet"
   Read height_feet
   height_meters = height_feet * meters
   IF height_meters < min_no THEN
      min_no = height_meters
   ENDIF
ENDFOR
Print "The height of the shortest Barcelona player is ",min_no," meters"
```

END PROCESSING

TESTING ALGORITHMS (Pg. 80)

- Before you can convert an algorithm to code, it is a good idea to check that it is complete and logical. You do this by carrying out the algorithm using actual values.
- There are two ways you can do this:

1. **DRY-RUN TESTING**
2. **TRACE TABLES**

DRY RUN TESTING (Pg. 80)

- A quick and easy way to test whether your algorithm works is to use a dry-run test, also called desk checking. This is where you substitute values for the variables and follow the instructions in the algorithm step by step to arrive at a solution. A dry-run test will tell you if there are any logic errors in your algorithm, because if a step is out of sequence, you will not be able to follow the calculation properly.

Here is part of the pseudocode:
Read b
Read c
a = b + c
d = a * 2 + c
e = d – 3
To conduct a dry-run test, substitute actual values for the variables that are entered, here b and c.
Assume b = 10 and c = 2.
a = 10 + 2 = 12
d = 12 * 2 + 2 = 26
e = 26 – 3 = 23

If an algorithm contains logic errors, you will not be able to complete a dry-run test.
Here is an example:
Read b
Read c
a = b + d
d = b * 2 + c
e = d – 3
Using the same values for b and c as the previous example,
the dry-run test would go as follows:
a = 10 + ?

TRACE TABLES (Pg. 81)

- Trace tables are useful if you have used loops in your algorithm. They are tables that track each variable as it progresses through the calculation. It will show you the output of each cycle of calculation within a problem. Here is an example:
- *Two numbers have starting values of 0 and 1 respectively. While the first number is less than 12 then add 3 to the first number and 2 to the second number. Print the results after each calculation.*

```
Number1 = 0
Number2 = 1
WHILE Number1 < 12 DO
   Number1 = Number1 + 3
   Number2 = Number2 + 2
   Print Number1, Number2
ENDWHILE
```

Number1	Number2
0	1
3	3
6	5
9	7`
12	9

TRACE TABLES (Pg. 81)

- The first step is to create a table with a column for each variable and one row per pass, so the number of rows depends on how many times the calculation is carried out. In this case you create a two-column table and you trace the loop until it can no longer be carried out. The following table shows the results
- The first row contains the initialized values. The second row contains the value of Number after 3 is added to it and the value of Number2 after 2 is added to it. The loop stops after Number reaches 12.