Ministry of Secondary Education
Progressive Comprehensive High School
PCHS Mankon – Bamenda
Department of **Computer Studies**

Republic of Cameroon
Peace – Work – Fatherland
School Year: 2014/2015

TOPIC: ALGORITHM AND FLOWCHART

Class: Comp. Sc. A/L **By:** DZEUGANG PLACIDE

The computers work on a set of instructions called ***computer program***, which clearly specifies the way to carry out a task. An analogy of this may be thought of as the instructions given by the manager or team leader to its team. The team members follow those instructions and accordingly perform their duties. Similarly, a computer also takes instructions in the form of computer programs to carry out the requested task. This chapter aims to

Learning objectives

After studying this lesson, student should be able to:

- Discuss what an algorithm is and how to use it to represent the solution of a problem.
- Use Flowchart and Pseudocode represent algorithms

Contents

I. ALGORITHM

I.1. Definition

Algorithms are one of the most basic tools that are used to develop the problem-solving logic. An **algorithm** is defined as a finite sequence of explicit instructions that when provided with a set of input values, produces an output and then terminates. Algorithm is not language specific. We can use the same flowchart to code a program using different programming languages. Many algorithms can be design for the same task.

I.2. Properties of an algorithm

Note that algorithms are not computer programs, as they cannot be executed by a computer. Some properties of algorithm are as follows:

- **Finiteness** – the algorithm stops after a finite number of instructions are executed.
- **Definiteness or unambiguous**: There must be no ambiguity in any instruction. This means that the action specified by the step cannot be interpreted in multiple ways & can be performed without any confusion.
- **Input**:- an algorithm accepts zero or more inputs
- **Output**:- it produces at least one output.
- **Effectiveness:**- it consists of basic instructions that are realizable. This means that the instructions can be performed by using the given inputs in a finite amount of time.

I.3. Example of algorithm

We use algorithms in our daily life. For example, to wash hand, the following algorithms may be used.

1. Start 2. Turn on water 3. dispense soap 4. wash hand till clean 5. Rince soap off 6. Turn off water 7. Dry hand 8. Stop	1. Start 2. Turn on water 3. dispense soap 4. Repeat Rub hands together 5. until hand clean 6. Rince soap off 7. Turn off water 8. Dry hand 9. Stop

The above-mentioned algorithm terminates after six steps. This explains the feature of finiteness. Every action of the algorithm is precisely defined; hence, there is no scope for ambiguity.

II. REPRESENTATION OF AN ALGORITHM

Algorithms can be represented in different ways:

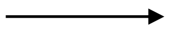
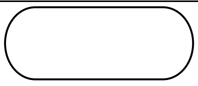
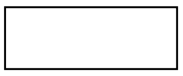
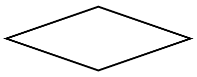
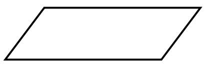
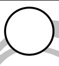
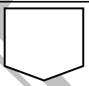
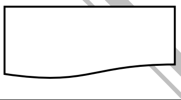
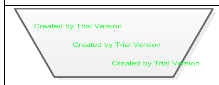
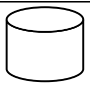
- By Flowchart
- By Pseudocode

II.1. Flowchart

Flowchart is the diagrammatic representation of an algorithm with the help of symbols carrying certain meaning. Using flowchart, we can easily understand a program.

a) Flowchart Symbols

Flowcharts use simple geometric symbols and arrows to define relationships. Some standard symbols that are frequently required for flowcharts are shown:

Symbol	Symbol name	Description
	Flow Lines	Flow lines are used to connect symbols. These lines indicate the sequence of steps and the direction of flow.
	Terminal	**The terminator symbol** represents the start points, end points, and potential outcomes of a path.
	Processing	used to illustrate a process, action or an operation. Examples: "Add 1 to X"; "save changes" or similar.
	Decision	These typically contain a Yes/No question or True/False test.
	Input/output	It represents data that is available for input or output. It may also represent resources used or generated. **Ex**: Get X from the user; display X.
	Connector	Connector symbol is used to join different flow lines.
	Off- page Connecter	This symbol is used to indicate the flowchart continues on the next page
	Document	Document is used to represent a paper document produced during the flowchart process.
	Manual Input	Manual input symbol represents input to be given by a developer / programmer.
	Magnetic Disk	This is used to represent data input or output from and to a magnetic disk.

b) Advantages of flowchart

The flowchart shows how the program works before you begin actually coding it. Some advantages of flowcharting are the following.

- **Communication**: Flowcharts are helpful in explaining the program to other people.

- **Effective analysis**: With the help of flowchart, the problem can be analysed more effectively
- **Proper documentation**: Program flowchart serves as a good program documentation, which is needed for various purposes.
- **Efficient coding**: Once the flowchart is drawn, it becomes easy to write the program in any high level language
- **Proper debugging**: The flowchart help in the debugging process
- **Efficient program maintenance**: The maintenance of operating program become easy with the help of flowchart

c) Limitations of Flowcharts

Flowchart can be used for designing the basic concept of the program in pictorial form, but cannot be used for programming purposes. Some of the limitations of the flowchart are given below:

- **Complex**: The major disadvantage in using flowcharts is that when a program is very large, the flowcharts may continue for many pages, making them hard to follow.
- **Costly**: If the flowchart is to be drawn for a huge program, the time and cost factor of program development may get out of proportion, making it a costly affair.
- **Difficult to Modify**: Due to its symbolic nature, any change or modification to a flowchart usually requires redrawing the entire logic again, and redrawing a complex flowchart is not a simple task.
- **No Update**: Usually, programs are updated regularly. However, the corresponding update of flowcharts may not take place, especially in the case of large programs.

II.2. Pseudocode

Pseudocode is a detailed yet readable description of what an algorithm must do, expressed in a formally-styled natural language rather than in a programming language. It describe the entire logic of the algorithm so that implementation becomes a rote mechanical task of translating line by line into source code.

a) Pseudocode Structures

Before going ahead with pseudocode, let us discuss some keywords, which are often used to indicate input, output and processing operations.

- Input: READ, OBTAIN, GET and PROMPT
- Output: PRINT, DISPLAY and SHOW
- Compute: COMPUTE, CALCULATE and DETERMINE
- Initialize: SET and INITIALIZE
- Add One: INCREMENT

b) Example of pseudocode

The pseudocode given below calculates the area of a rectangle.

Pseudocode: To calculate the area of a rectangle

```
START
PROMPT the user to enter the length of the rectangle
PROMPT the user to enter the width of the rectangle
COMPUTE the area by multiplying the length with width
DISPLAY the area
STOP
```

c) Advantages of Using Pseudocode

Some of the most significant benefits of pseudocode are as follows:

- It is easier to develop a program from a pseudocode rather than from a flowchart or decision table.
- Often, it is easy to translate pseudocode into a programming language, a step which can be accomplished by less-experienced programmers.
- Unlike flowcharts, pseudocode is compact and does not tend to run over many pages. Its simple structure and readability makes it easier to modify as well.
- Pseudocode allows programmers who work in different computer languages to talk to each other.

d) Disadvantages of Using Pseudocode

Although pseudocode is a very simple mechanism to simplify problem-solving logic, it has its own limitations. Some of the most notable limitations are as follows:

- It does not provide visual representation of the program logic.
- There are no accepted standards for writing pseudocodes. Different programmers use their own style of writing pseudocode.
- It is quite difficult for the beginners to write pseudocode as compared to drawing a flowchart.

III. ALGORITHM STRUCTURES

The 'structured' part of pseudocode and flowchart is a notation for representing three general **programming constructs**: ***sequence, selection*** and ***repetition***. Each of these constructs can be embedded inside any other construct. It has been proven that three basic constructs for flow of control are sufficient to implement any 'proper' algorithm.

- **Sequence**, where information can flow in a straight line.
- **Selection** (branched), where the decisions are made according to some predefined condition.

- **Repetition**, where the logic can repeat in a loop, that is, where a sequence of steps is repeated until the desired output is obtained.

III.1 Sequence

Sequence construct is a linear progression where one task is performed sequentially after another. The actions are performed in the same sequence (top to bottom) in which they are written

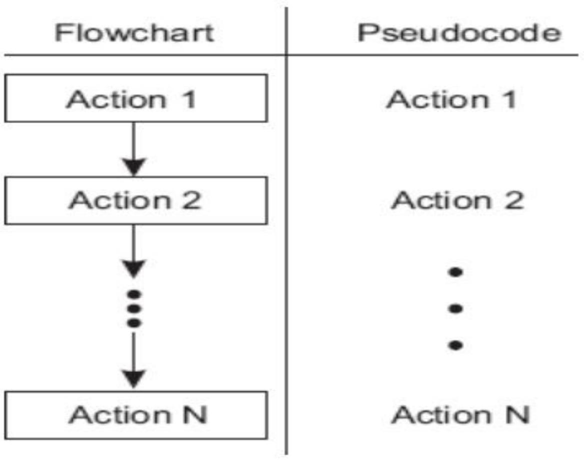
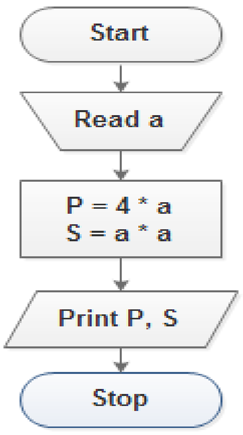


Fig. Flowchart and Pseudocode for Sequence Construct

Example

Write an algorithm and flowchart for calculating the perimeter and surface of square, if the default length of the sides of the square is a.

Flowchart	Pseudocode	C

 Start
Read a
P = 4 * a
S = a * a
Print P, S
Stop
	Begin
	Input a
	P = 4 × a
	S = A × a
	Print P, S
	END
		#include<stdio.h>
		int main()
		{
		int a, P, S;
		printf("Enter the length");
		scanf("%d",&a);
		P = 4 * a ;
		S = a * a ;
		printf("\nPerimetre = %d", P);
		printf("\nSurface = %d", S);
		return 0 ;
		}

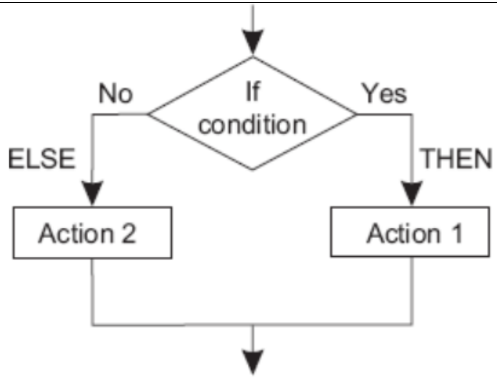
Note that there is no branching and no process is repeated again. Each process is contributing something to the next process.

III.2 Selection (Decision)

Selection is a process of deciding which choice is made between two or more alternative courses of action. Selection logic is depicted as an IF-THEN-ELSE-ENDIF or CASE-ENDCASE structure. As the name suggests, in case of the IF-THEN-ELSE-ENDIF construct, if the condition is true, the true alternative actions are performed and if condition is false, then false alternative actions are performed on.

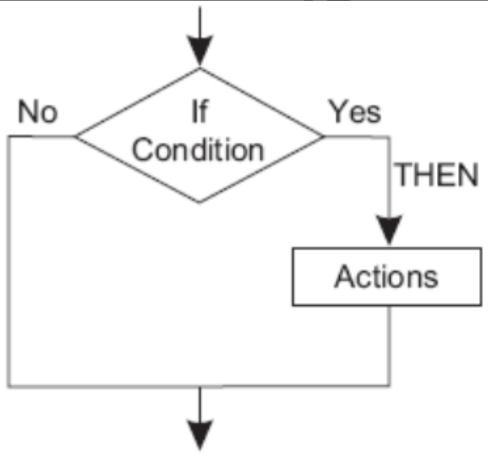
a) IF-THEN-ELSE-ENDIF construct

Flowchart	Pseudocode	C

 If condition
No
Yes
ELSE
THEN
Action 2
Action 1
	•
	•
	•
	IF condition THEN
	List of actions
	ELSE
	List of different actions
	ENDIF
	•
	•
	•
		•
		•
		if(condition)
		{
		List of actions
		}
		Else
		{
		List of different actions
		}
		•
		•

Note that the ELSE keyword and 'Action 2' are optional. In case you do not want to choose between two alternate courses of actions, then simply use IF-THEN-ENDIF

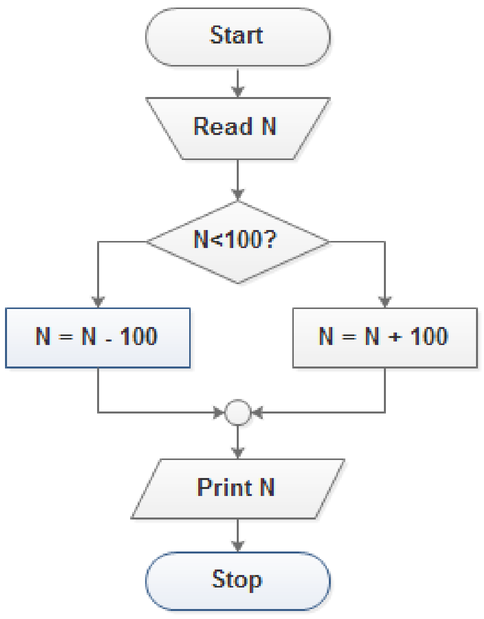
Flowchart	Pseudocode	C

	•
	•
	•
	IF condition THEN
	List of actions
	ENDIF
	•
	•
	•
		•
		•
		•
		if(condition)
		{
		List of actions
		}
		•
		•
		•

Hence, if the condition is true, then perform the list of actions listed in the IF-THEN-ENDIF construct and then move on to the other actions in the process. In case the condition is false, then move on to the rest of the actions in the process directly. Let us write a pseudocode to find the largest of three numbers

Example: Write an algorithm and flowchart which a given number **N** *increased* by 100 if **N** is less than 100, otherwise **N** is *decreased* by the 100. Print this result.

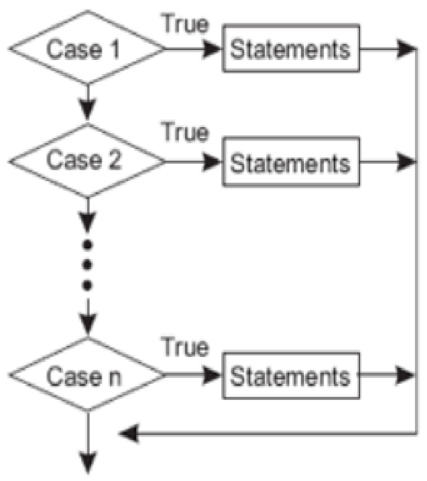
Flowchart	Pseudocode	C

 Start
Read N
N<100?
N = N - 100
N = N + 100
Print N
Stop
	START
	  Input N
	  If N < 100 Then
	    N = N × 100
	  Else
	    N = N - 100
	  Print N
	STOP
		#include<stdio.h>
		int main()
		{
		  int N;
		  printf("Enter the number: ");
		  scanf("%d",&N);
		  if(N<100)
		    N=N+100;
		  Else
		    N=N-100;
		  printf("\nnumber is %d",N);
		return 0;
		}

b) CASE-ENDCASE construct

If there are a number of conditions to be checked, then using multiple IFs may look very clumsy. Hence, it is advisable to use the CASE-ENDCASE selection construct for multipleway selection logic. A CASE construct indicates a multiway branch based on many conditions. CASE construct uses four keywords, CASE, OF, OTHERS and ENDCASE, along with conditions that are used to indicate the various alternatives.

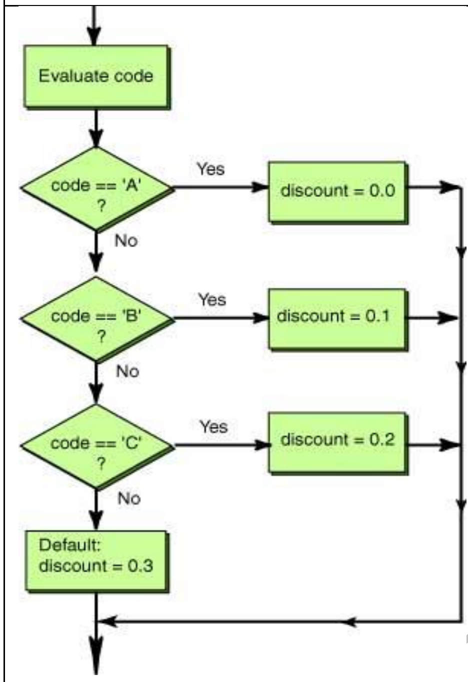
Flowchart	Pseudocode	C

 Case 1
True
Statements
Case 2
True
Statements
Case n
True
Statements
	CASE expression OF
	  Condition 1: Sequence 1
	  Condition 2: Sequence 2
	  •
	  •
	  Condition n: Sequence n
	  OTHERS : default sequence
	ENCASE
		case (expression)
		{
		  case value 1:
		    Sequence 1;
		    break ;
		  case value 2:
		    Sequence 2;
		    break ;
		  •
		  •
		  case value n:
		    Sequence n;
		    break ;
		  default :
		    default sequence ;
		}

CASE construct performs the same process as multiple IFs, but it is much easier to read and write. Conditions are normally numbers or characters indicating the value of 'Expression'

Example: ***To assign discount according to the code***

Flowchart	Pseudocode	C

 Evaluate code
code == 'A' ?
Yes
discount = 0.0
No
code == 'B' ?
Yes
discount = 0.1
No
code == 'C' ?
Yes
discount = 0.2
No
Default: discount = 0.3
	START
	READ code
	CASE Grade OF
	A : discount = 0.0
	B : discount = 0.1
	C : discount = 0.2
	OTHERS : discount = 0.3
	ENDCASE
	DISPLAY discount
	STOP
		#include<stdio.h>
		int main()
		{
		char code ;
		switch (code)
		{
		case 'A': discount = 0.0;
		break;
		case 'B': discount = 0.1;
		break;
		case 'C': discount = 0.2;
		break;
		default: discount = 0.3;
		}
		Printf("discount is: %f ", discount);
		}

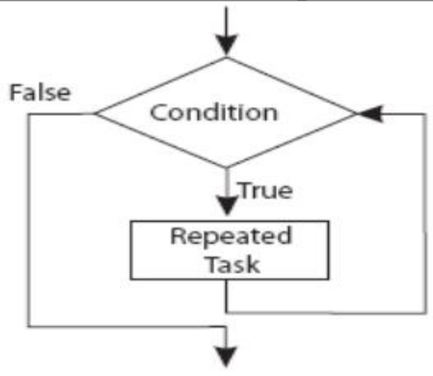
III.3 Repetition (Looping)

Looping construct is used when some particular task(s) is to be repeated for a number of times according to the specified condition. By using looping, the programmer avoids repeating the same set of instructions. As the selection, the loop is also represented in flowchart by a diamond. The difference is just at the orientation of the arrows.

a) WHILE- ENDWHILE

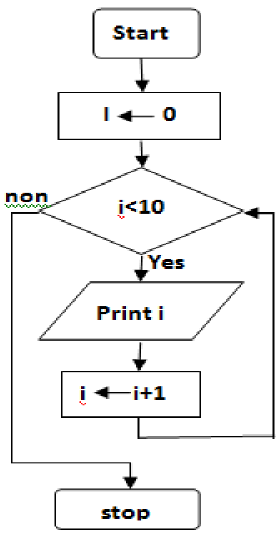
In case of WHILE-ENDWHILE, the loop will continue as long as the condition is true. The loop is entered only if the condition is true. The 'statement' is performed for each iteration. At the conclusion of each iteration, the condition is evaluated and the loop continues as long as the condition is true.

Flowchart	Pseudocode	C

	WHILE condition is True
	statements
	ENDWHILE
		while (condition)
		{
		statements
		}

Example: ***To display the first ten natural numbers using DO WHILE-ENDDO***

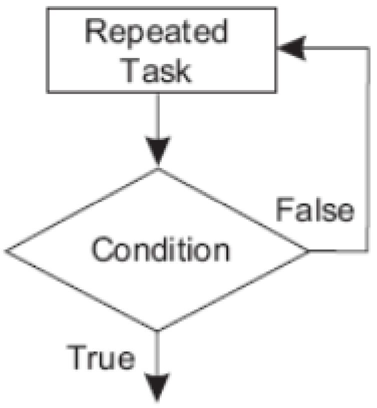
Flowchart	Pseudocode	C

 Start
I ← 0
non
i<10
Yes
Print i
i ←i+1
stop
	INITIALIZE Count to zero
	WHILE Count >= 10
	ADD 1 to Count
	PRINT Count
	ENDWHILE
	STOP
		`#include<stdio.h>`
		`main()`
		`{`
		`    int i=0;`
		`    while(i<10)`
		`    {`
		`       printf("%d ",i);`
		`       i++;`
		`    }`
		`    return 0;`
		`}`

b) REPEAT-UNTIL

The REPEAT-UNTIL loop is similar to the WHILE-ENDWHILE, except that the test is performed at the bottom of the loop instead of at the top

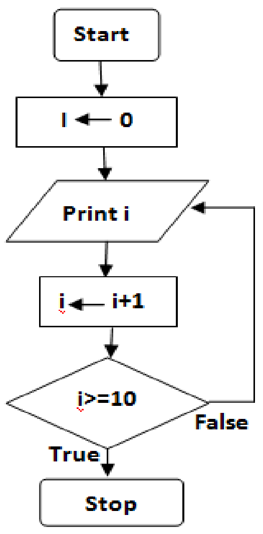
Flowchart	Pseudocode	C

	Repeat
	Statements
	Until condition is false
		Do
		{
		
		}
		While (condition)

The 'statement' in this type of loop is always performed at least once, because the test is performed after the statement is executed. At the end of each iteration, the condition is evaluated, and the loop repeats until the condition gets true. The loop terminates when the condition becomes true.

Example To display the first ten natural numbers using REPEAT-UNTIL

Flowchart	Pseudocode	C

 Start
I ← 0
Print i
i ← i+1
i>=10
False
True
Stop
	INITIALIZE Count to zero
	REPEAT
	ADD 1 to Count
	PRINT Count
	UNTIL Count is less than 10
	STOP
		`#include<stdio.h>`
		`main()`
		`{`
		`    int i=0;`
		`    do`
		`    {`
		`       printf("%d ",i);`
		`       i++;`
		`    }`
		`    while(i<10);`
		`    return 0;`
		`}`

APPLICATION EXERCISES

Exercise 1: Write the flowchart corresponding to the following pseudo code

Pseudocode 1	Pseudocode 2	Pseudocode 2

Start
num1 = 5
num2 = 10
num3 = 15
sum = num1 + num2 + num3
average = sum/3.0
print average
Stop
	Set total to zero
	Set grade counter to one
	While grade counter is less than or equal to ten
	Input the next grade
	Add the grade into the total
	Set the class average to the total divided by ten
	Print the class average.
		initialize passes to zero
		initialize failures to zero
		initialize student to one
		while student counter <= to ten
		input the next exam result
		if the student passed
		add one to passes
		else
		add one to failures
		add one to student counter
		print the number of passes
		print the number of failures
		if eight or more students passed
		print "raise tuition"

Exercise 2: Create flowcharts to represent these short tasks:

a. "If it's raining, bring an umbrella."
b. "Take twenty paces, then turn and shoot."
c. "Go forward until the Touch Sensor (on port 1) is pressed in, then stop."
d. "Follow Liberty Avenue for 2 miles, then take a left turn onto 40th Street. Go until you reach the bridge, but don't cross the bridge. Instead, make a right turn onto Foster Street, then take the first left turn. Follow that road until you reach the National Robotics Engineering Consortium building."
e. "Turn on oven. Cook turkey for 4 hours or until meat thermometer reaches 180 degrees."

Exercise 3: Make the flowchart, and write the pseudocode for the following problem:

Given a Fahrenheit temperature, calculate and display the equivalent centigrade temperature. The following formula is used for the conversion: **C = 5 / 9 * (F – 32)** where F and C are the Fahrenheit and centigrade temperatures.

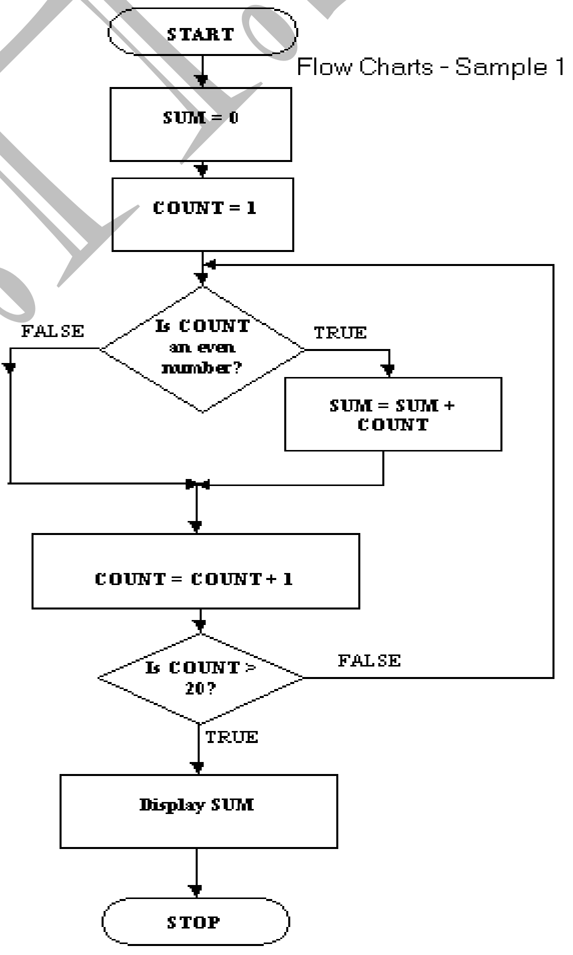
Exercise 4:

On a separate sheet of paper, make a flowchart organizing the "flow" of getting ready to go to school in the morning. Be sure to include the following steps in your chart, but don't be afraid to add other things if you need them!

Select something to wear	*Look for your shoes*	*Put your shoes on*
Take a shower	*Brush your teeth*	*Hit snooze button*
Eat breakfast	*Put toast in the toaster*	*Get dressed*
Leave house for school	*Check your alarm clock*	*Comb your hair*
Get out of bed	*Turn on shower*	*Check the time*

Exercise 5: Let's consider the following flowchart

- **a)** Identity the programming construct available in the flowchart and explain how each of them works.
- **b)** Do the dry run of the flowchart
- **c)** What does the flowchart do?
- **d)** Write the corresponding pseudocode of the flowchart


START
Flow Charts - Sample 1
SUM = 0
COUNT = 1
Is COUNT an even number?
FALSE
TRUE
SUM = SUM + COUNT
COUNT = COUNT + 1
Is COUNT > 20?
FALSE
TRUE
Display SUM
STOP