COMPUTER SCIENCE

S4MCE

UNIT 8

CONTROL STRUCTURES AND ONE DIMENSION ARRAY

Introduction

Control structures are statements or symbols used in algorithms to represent the logical flow or order in which program statements are to be executed

Main key to be discussed:

- Conditional logic
- Control structures
- One-dimensional array

Conditional logic

"If I have money, I will buy a car". This statement is a conditional logic means that certain conditions must be satisfied for an action to be taken.

Therefore, a conditional logic is a proposition formed by combining two or more facts using the words like **if**, **case** and **then**. The conditions in the if statement are combined using logical links like: **and**, **or** and **not**.

Simple conditional logic

Simple conditional logic requires only one fact for an action to be taken, hence statements do not require use of logical links like and, or and not.

Example: Teacher promises that if "you participate in class", then "you will get five extra points"

- **Fact**: you participate in class – can be true/false
- **Action**: you will get extra five points - can be true/false

Compound conditional logic

Compound conditional logic use of logical links to combine several facts for an action to be taken.

For example, the following statement requires two conditions to be fulfilled for the teacher to take action:

• The teach promises that if "you are always punctual"" **and** "participates in class" **then** "you will get five extra points"

Conditions linked with **AND** logic requires an action to be taken only when all conditions are true

p	q	P AND q
1	1	1
1	0	0
0	1	0
0	0	0

Conditions linked with an **OR** logic lead to an action when either **one** or **both** are **true**.

For example, the teacher may decide to award five points if a student is punctual or participates in class. This statement can be represented using OR logic below:

p	q	P OR q
1	1	1
1	0	1
0	1	1
0	0	0

pseudocode that shows a use of conditional logic to check if a person is an adult:

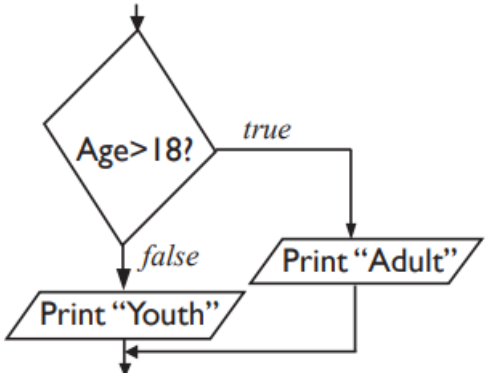
IF Age > 18 THEN

PRINT "Adult "

ELSE

PRINT "Youth "

END IF



Control Structures

Control structures (statements) is conditional logic that determines the flow of an algorithm or execution of a program.

Types of control structures

- Sequence
- Selection
- Looping

Sequence Control Structure

Sequence control structure is the logical flow of statements one after another in the order in which they are written.

sequence control in which two numbers are first entered before sum and product are calculated and displayed on the screen

BEGIN

SET variables sum,product,num1,num2

PRINT "Enter 2 numbers"

READ number1, num2

sum = number1 + num2

product = num1 * num2

PRINT sum, product

END

Selection Control Structure

Selection control structure/ decision control statement is a conditional logic used when there is one or more alternatives to choose from.

Types of selection control structure

- **If ...then**
- **If...else**
- **Nested if**
- **Switch/case**

If … then selection

If…then is a conditional logic used to test whether the condition is true before an action is taken.

Syntax is:

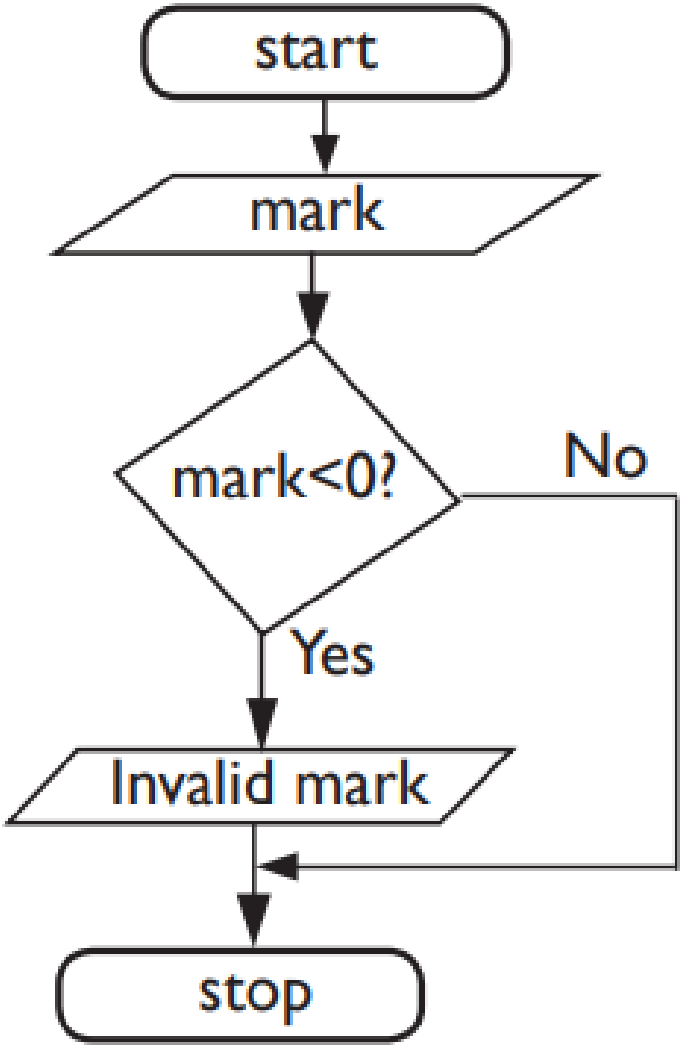
If condition is true then

Do Task-A

For example: a flowchart with if … then selection used to test whether a number entered is less than zero. If the number is negative, the algorithm displays invalid mark

Explanation

1. Once the user enters a mark, the algorithms checks whether the input is less than zero.
2. If true then statement 'invalid mark' is displayed, otherwise nothing happens



If … else selection

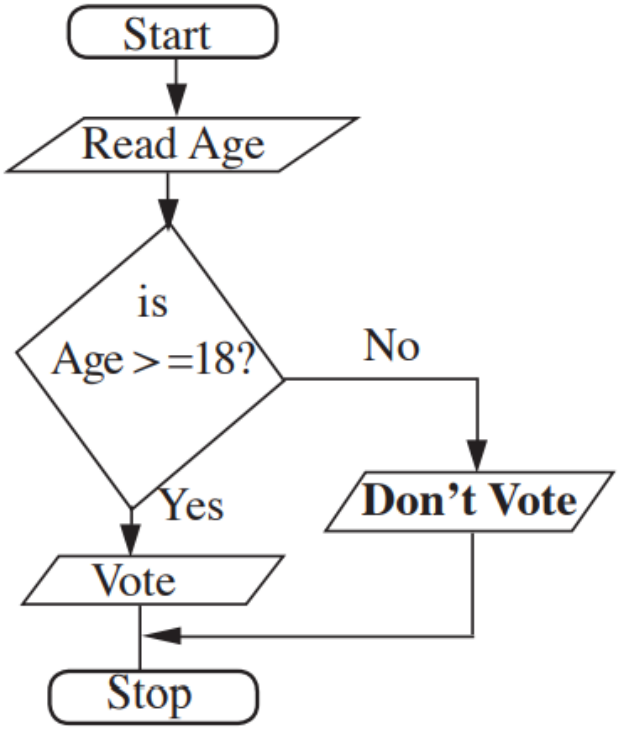
If … else selection is suitable when there are two available options.

Syntax:

IF THEN Statement 1

ELSE Statement *2*

END IF



BEGIN
USE VARIABLE: AGE AS INTEGER **WRITE** "Enter one's age"
READ Age
IF Age > =18
WRITE "Vote"
ELSE
WRITE "Don't Vote"
END

Nested IF

Nested IF selection is used where several options have to be considered to make a selection

Syntax:

IF<boolean expression>THEN

statement 1

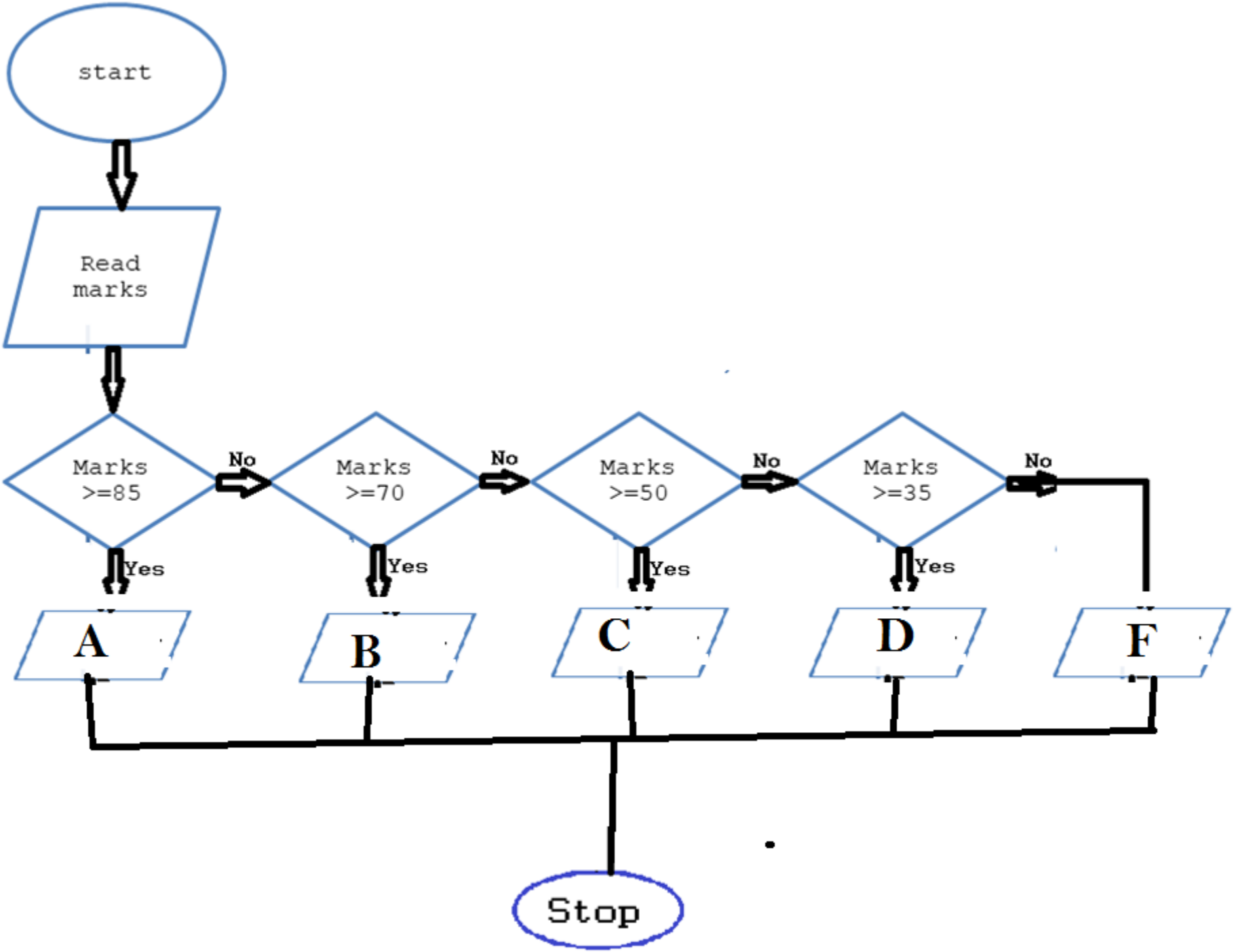
ELSE IF<boolean expression> THEN statement 2

ELSE IF<boolean expression>THEN statement 3

ELSE

statement 4

END IF


start
Read marks
Marks >=85
No
Marks >=70
No
Marks >=50
No
Marks >=35
No
Yes
Yes
Yes
Yes
A
B
C
D
F
Stop

Switch/Case selection

An alternative to nested if selection is use of switch/case selection control. The following algorithm represents the general syntax of a switch statement

SWITCH(expression)

CASE expression 1:

statement 1

statement 2

.

.

CASE expression n:

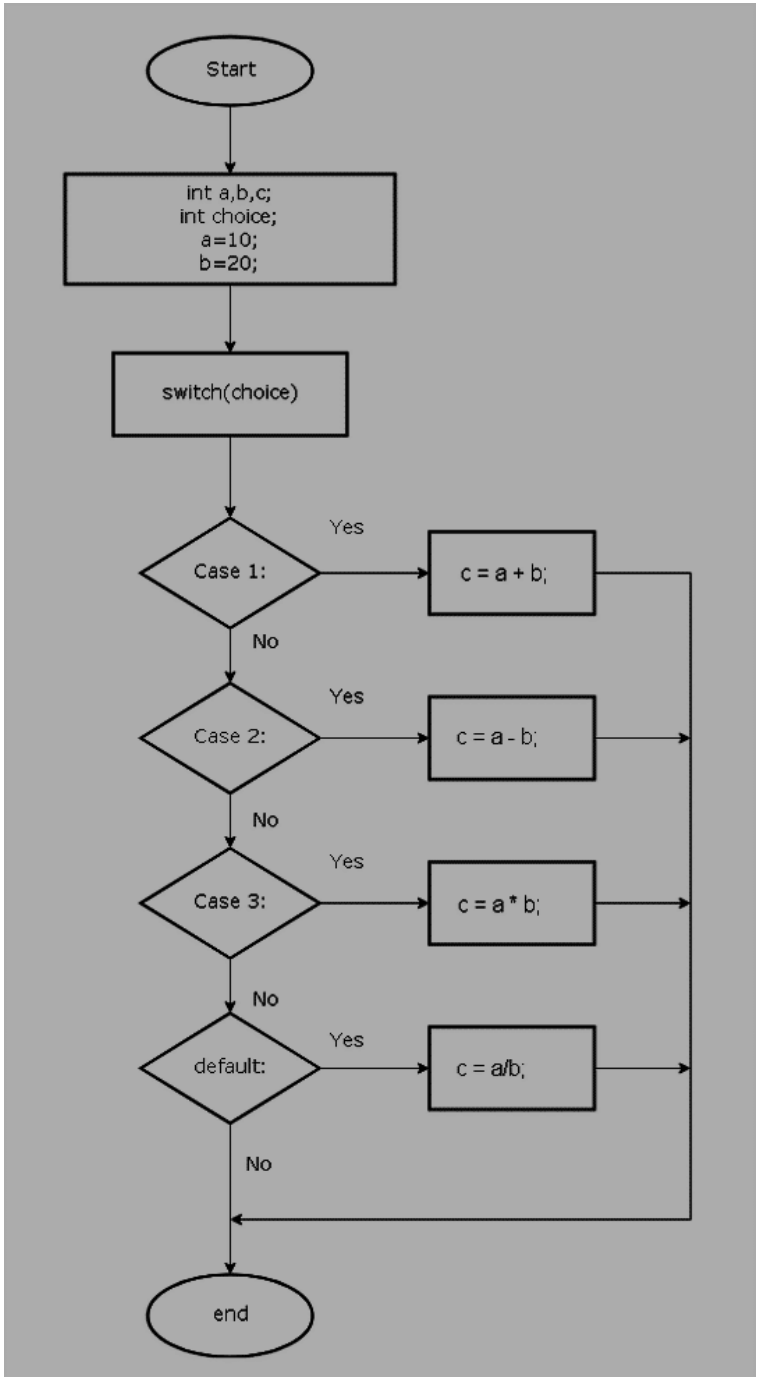
statement(s)

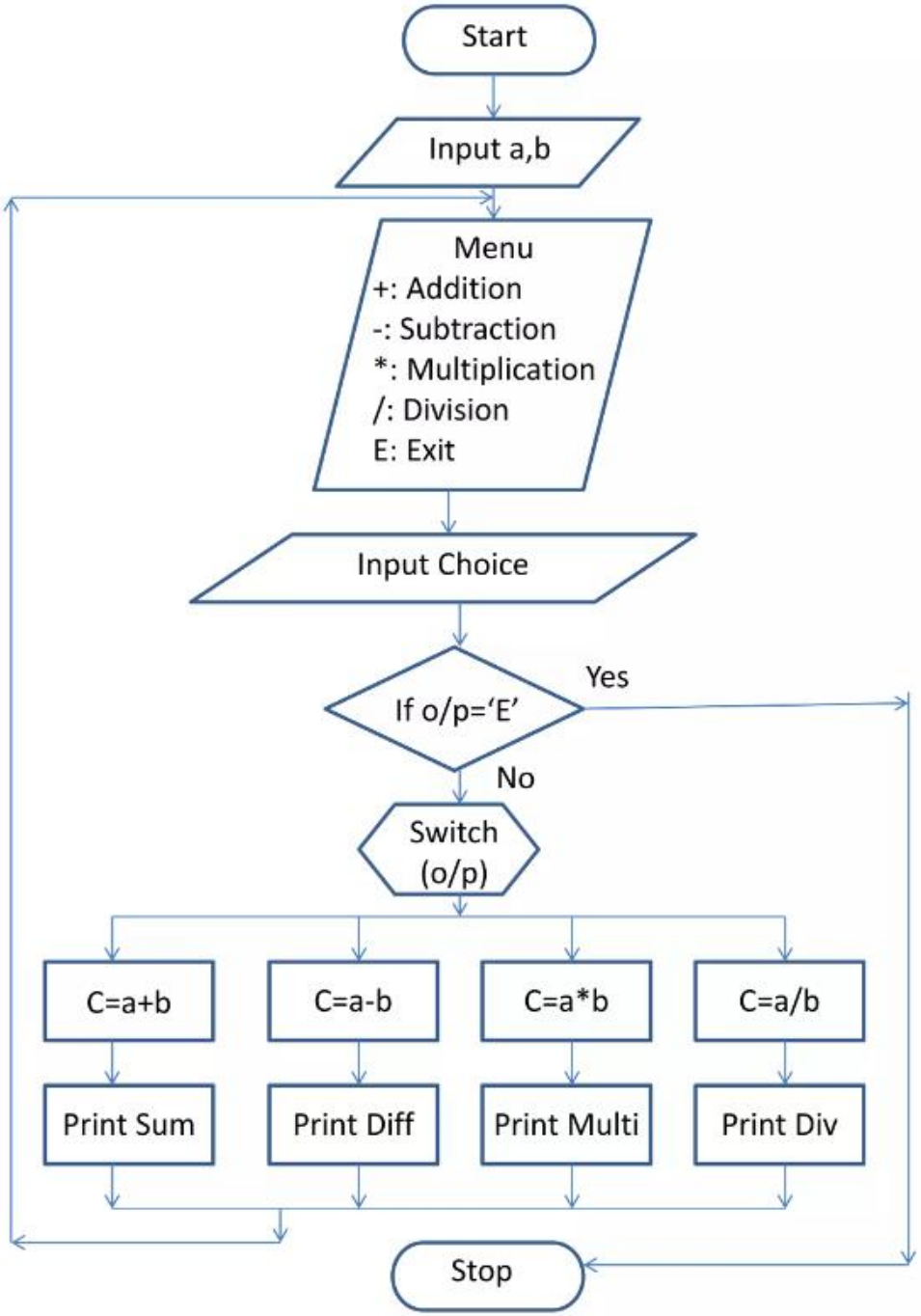
DEFAULT: statement(s);

END SWITCH

Sample Switch...Case Selection

```
BEGIN
use variable number AS Integer
PRINT "Enter menu item"
READ number;
SWITCH(number)
CASE of 1:
PRINT "My choice is Milk"
CASE of 2:
PRINT "My choice is Tea"
CASE of 3:
PRINT "My choice is Coffee"
DEFAULT:
PRINT "Your choice is not valid"
END SWITCH
END
```


Start
int a,b,c;
int choice;
a=10;
b=20;
switch(choice)
Case 1:
Yes
c = a + b;
No
Case 2:
Yes
c = a - b;
No
Case 3:
Yes
c = a * b;
No
default:
Yes
c = a/b;
No
end


Start
Input a,b
Menu
+: Addition
-: Subtraction
*: Multiplication
/: Division
E: Exit
Input Choice
If o/p='E'
Yes
No
Switch (o/p)
C=a+b
C=a-b
C=a*b
C=a/b
Print Sum
Print Diff
Print Multi
Print Div
Stop

Looping control structure

It is a control structure that causes the program to repeatedly execute statements within the loop until the condition is false.

a. **FOR Loop:** this looping statement evaluates a condition before executing statement in the body of the loop.

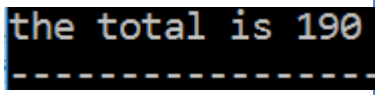
syntax:

FOR variable = lower_limit TO upper_limit DO

statements;

END FOR

Example: sum of first 20 natural numbers.



BEGIN

SET Sum = 0

FOR number=0 To number<20 Do

Sum=Sum+number

END FOR

PRINT Sum

END

b. WHILE Loop: it first evaluates the condition before executing the body of the loop.

Syntax:

WHILE < boolean expression> DO

statements

END WHILE

Example: Decrementation

```
BEGIN
SET num;
WHILE NUM>0 DO
num--
END WHILE
END
```

c. Repeat ...Until Loop: it is similar to the while loop except that the statement is executed at least once.

Example:

BEGIN

SET Number, Quotient, Remainder

SET Number=0, Quotient=0, Remainder=0

PRINT "Please enter a decimal number"

READ Number;

REPEAT Quotient = Number Div 2

Remainder = Number Mod 2

PRINT Remainder

Number = Quotient

UNTIL Number=0

PRINT "Read remainder upwards";

END.

Finite and Infinite Loops

A finite loop repeatedly executes a set of instructions until a specific condition is met.

An infinite loop (endless loop) continues looping indefinitely due to a condition that is never met

For example: an infinite loop in which the value of x is reset to 1 hence the condition x< 5

```
DO x = 1
x = x + 1
PRINT x
END WHILE
```

BREAK AND CONTINUE STATEMENTS

A break statement is used to force immediate exit from a loop or selection statements

```
BEGIN
FOR count=1 TO 10 DO
IF count = 5
THEN break
END IF
PRINT count //1,2,3,4
END FOR
END
```

Continue statement is used in looping to skip the remaining statements in the body of the loop and perform the next iteration

```
BEGIN
FOR count=1 TO 10 DO
IF count = 5
THEN continue
END IF
PRINT count
END FOR
END
```

Exercise

1. Differentiate between selection and iteration control structures.

Selection branches to an alternative action if while **iteration** repeats existing statement for a finite number of times

2. Explain the importance of the following selection statements:

(a) IF..THEN: is used to execute statements under true block if the Boolean return true

(b) Nested IF is used to test multiple options

(c) SWITCH is used when instead of using nested if to test multiple conditions, we use case

3. Explain three types of looping control structures.

Three types looping control structures are **while**, **repeat..until** and **for loop.**

4. Design an algorithm for a program that would be used to compare three numbers x, y and z, and then display the least among the three.

```
BEGIN
  PRINT  Enter x, y and  z
  READ x, y, z;
     IF  x< y Then
        IF  x<z   = Then;
           least = x
         ELSE
           least = z
     ELSE
       IF y<z Then
           least = y
     ELSE
           least = z
     ENDIF
    PRINT least
END.
```

5. State four types of selection control structures supported by most structured programming languages.

if, if..else, nested if and switch ..case selection

The break statement

- It is used to terminate a loop immediately without checking the test expression.
- It can be used inside any loops such as:
  - ✓For loop
  - ✓While loop
  - ✓do while loop

The continue statement

- It is used to skip the current iteration of the loop and continue with the next iteration.
- It can be used with:
  - ✓For loop
  - ✓While loop
  - ✓do while loop

Go to statement allows you to transfer program execution to a different part of the code.

Example:

BEGIN

Repeat-again:

PRINT Please type your password:

READ mypassword

IF mypassword = secret

THEN

PRINT "login successful" ELSE

PRINT "incorrect password"

goto

Repeat-again

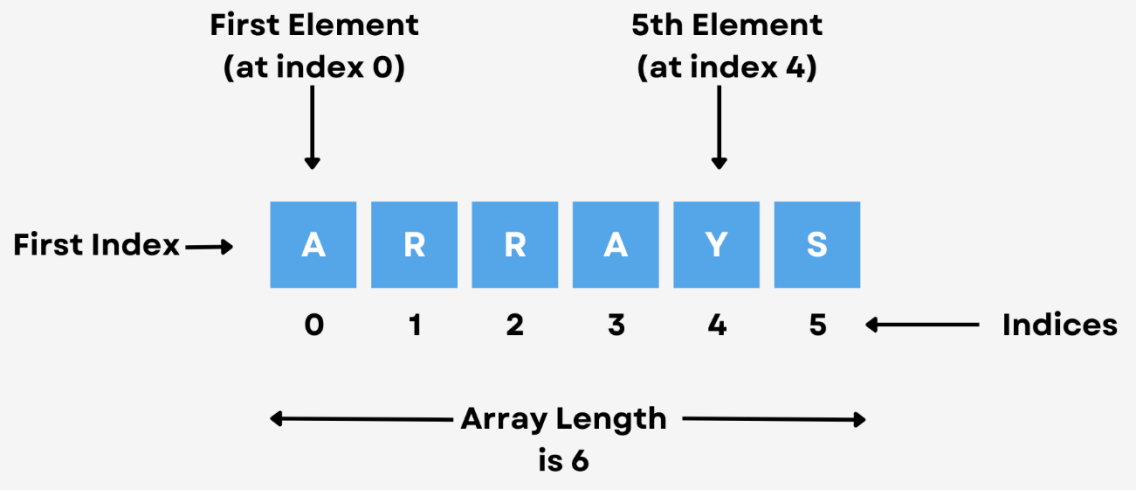
END

The Exit Statement is a statement used to terminate the program.

One-Dimensional Array

Array is a collection of similar type elements

A one-dimensional array is a group of elements having the same data type which are stored in a linear arrangement under a single variable name



Note: All the elements of a one-dimensional array are stored at contiguous memory locations. This means the elements are placed one after the other in memory, allowing for easy access and manipulation by using the **index/ subscript** of each element of the array.

Declaration and initialization of Arrays

To declare and initialize an array, some factors are considered:

- **Array name**: indicates the set of elements to be stored
- **Data type of elements**: type of elements to be stored in array.
- **Size of array**: the maximum number of elements to be stored in an array
- **Dimension**: it is the number of indices needed to select an element within the array

1. Array declaration

To declaring an array is the same as declaring other variables only that a computer reserves contiguous memory locations enough to store the number of elements

Syntax:

Array_name: Array [size] of datatype;

Example:

Scores: Array[10] of integer;

2. Array initialization

To initialize an array is to assign an initial value to array during declaration

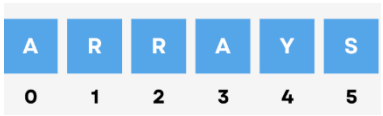
Syntax:

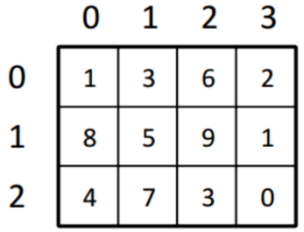
Array_name: Array[size]={value_1,value_2, …,value_n};

Example:

Scores: Array [10] = {34, 20, 45, 87, 92, 21, 43, 56, 12, 15};

One dimensional array





Multi-dimensional array/matrix

Accessing Array Element

In arrays, an element can be accessed by specifying the array name and the location (index) of the element. For example, to access the first element **(index 0)** in an array named scores, use **scores [0].** Once you access the element, you can then read or write a value into it.

- **Reading array elements**

To store (read) a value into an array, you need to know the name of the array and the index of the element

Example: to read a sing element in an array of elements. The value in **Scores[4]** may be read by using **READ Scores [4]**

The above function stores a value in the **fifth location** of the score array

Multiple values may be read into several elements using a **FOR loop**:

```
BEGIN
SET Scores=Array[10]of Integer
FOR Index=0 TO 9 DO
READ Scores[Index];
Index = Index + 1
LOOP
END FOR
END.
```

❑ Writing array elements

To write (display) elements from an array, use a **write** function together with the **array_name** and location (**index**) of the element.

Example:

the value in **Scores [1]** may be displayed by using **PRINT Scores[1]** .

To display multiple values such as the 10 elements in the Scores array, use the FOR loop by setting the initial value to 0 and the upper limit to 9

```
BEGIN
SET Scores=Array[10]of Integer
FOR Index=0 TO 9 DO
PRINT Scores[Index];
Index = Index + 1
LOOP
END FOR
END
```