2. Develop program in c language using conditional & loop control statement

- **Instruction** :- Program statements are called instruction.

Types of instruction :-

i) Data type declaration instruction
ii) I/O instruction
iii) Arithmatic instruction
iv) Control instruction

Control Instruction

→ If we want to control the sequance of operation of instruction, we have to give control instruction.

→ Control instructions are classified as -

i) Decision control instruction
ii) Iterative control instruction.
iii) Switch case control instruction
iv) Goto control instruction

Decision control Instruction / Selection control instruction

When we want to execute different instruction in different situation, we used decision control instruction

This can be implemented in c using -

(i) if statement (ii) if-else statement (iii) Conditional operator

(1) if statement

→ The keyword if tells the compiler that it is a decision control instruction.

→ Syntax :-

```
if (expression)
  statement ;
```

→ The expression written after if in brackets can a relational, arithmatic expression.

→ If the expression found is true then statement below it is executed. if we want to execute a set of statement then syntex should be as under.

```
if (expression)
{ statement 1 ;
  statement 2;
}
```

Note :- In case we use an arithmatic expression, all non-zero results are considered true.

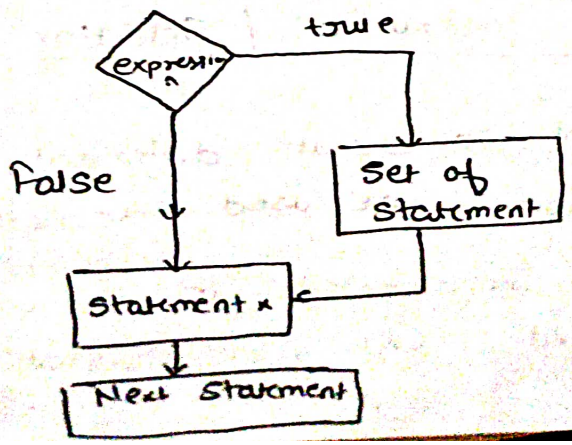


fig. - flow chart of simple if control

if - else statement

→ The if-else statement is extension of simple if statement.

→ In simple if statement we evaluate the expression and execute the statements when expression is true in if-else statement we can also evaluate execute a different set of statement when expression is false

→ Syntax :-

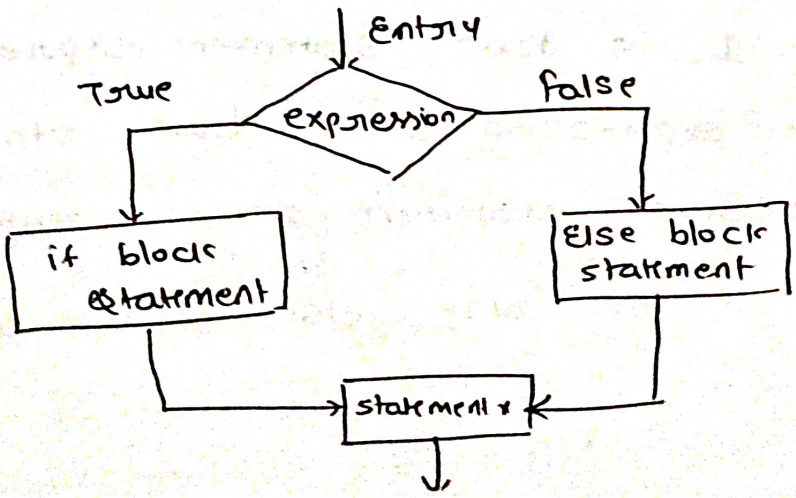
```
if (expression)
{   if block statement(s); }
else
{   else block statement (s) }
```

→ Semicolon is not used after if and else statement because if we will put semicolon compiler will not consider if block as its statement.

→ No condition is required after else

→ Else should be used just after if block.

→ Single else is not possible as it has no expression



flow - chart for if-else control

- **Nesting of if-else statements / Nested if-elses**

→ When a series of decision is required, nested if else is used.

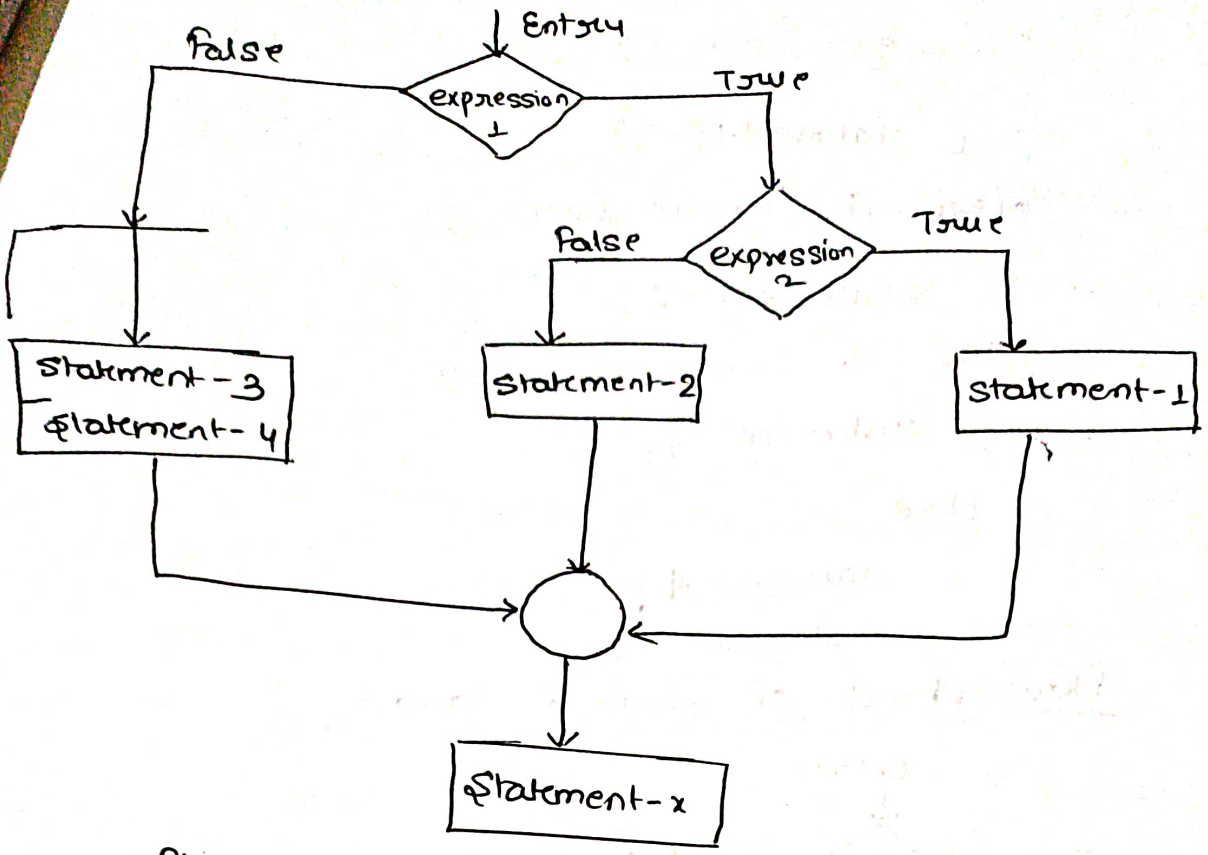
Syntax:-

```
if (expression-1)
  { if (expression-2)
      { statement-1;
      }
    else
      { statement-2;
      }
else
  { statement-3;
    statement-4;
  }
  statement-x;
```

} else block

→ if expression-1 is true ~~statement~~ expression-2 is evaluated, if expression-2 is true statement-1 is executed otherwise statement-2 is executed.

→ if expression-1 is false else block is executed.



Flow chart of nested if-else statement

Note :- else is always pair with most recent unpaired if

→ for better understanding of program make a habbit of indentation.

- The else-if ladder :-

→ While making a series of decision, there occurs a chain of if in which the statement associated with else is an if. That can be presented in below given form which is known as else if ladder.

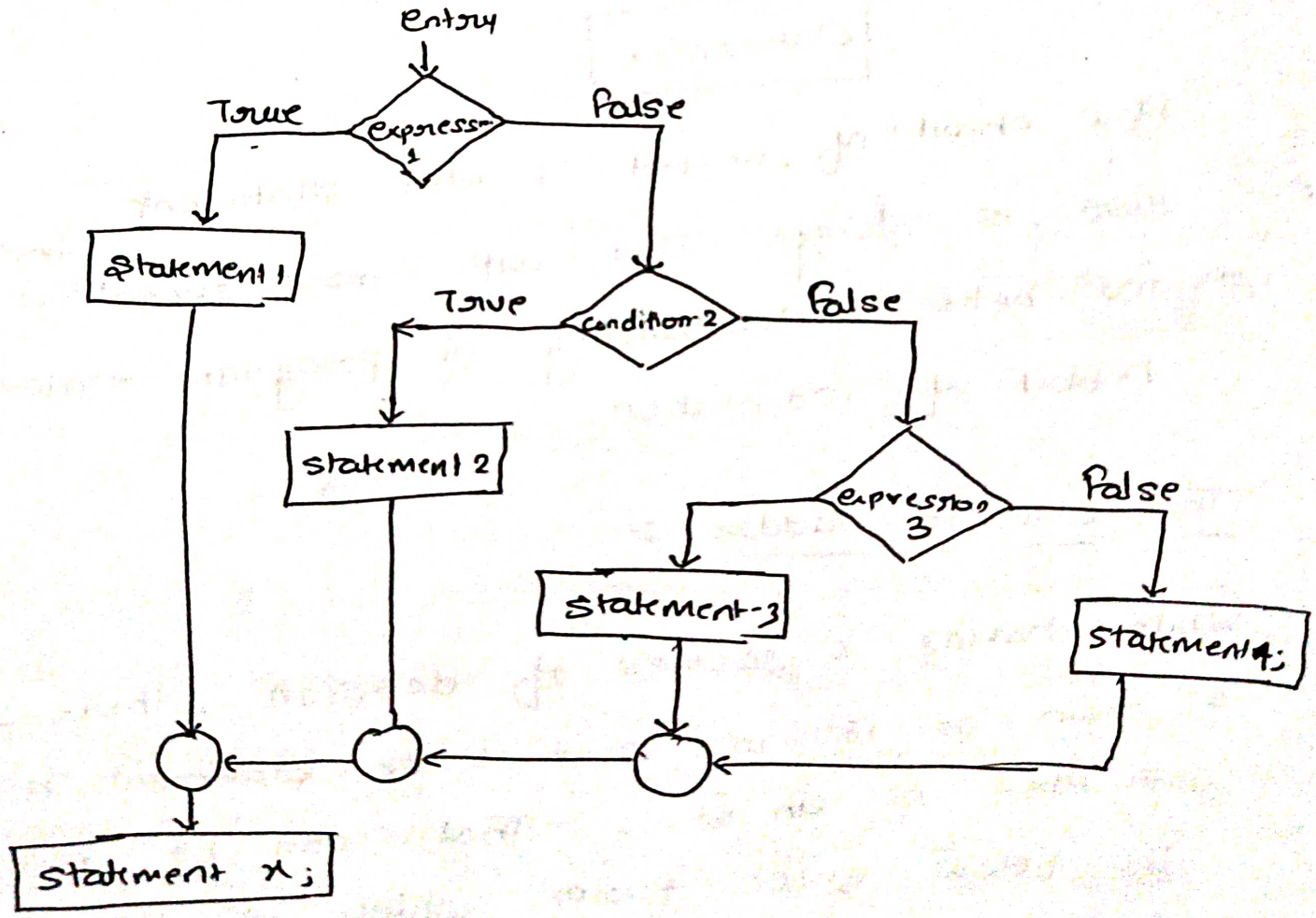
→ This reduces ~~id~~ indentation.

→ if 1st expression is true. Other expression need not to be checked

Syntax:-

```
if (expression-1)
    { statement-1(s);}
elseif if ('expression-2)
    statement-2;
else if (expression-3)
    statement-3;
else
    statement 4;
    statement x;
```

flow chart of else-if ladder



The ? : Operator

→ The conditional operator ? : sometimes called ternary operator since they take three argument.

→ The general form is

x = expression-1 ? expression-2 : expression-3 ;

Here expression-1 is evaluated, if it is true (non-zero), then expression 2 is evaluated and its value is assigned to x and if expression-1 is not true (zero), then expression-3 is evaluated & result is assigned to x.

eg ⇒ { int a=10, b=5;

x = (a>b) ? a : b; }

Result : x = 10

This can also be achieved using if else statement

```
{ int a=10, b=5;
  if (a>b)
  { x=a;
  else
      x=b;
}
```

The switch statement

→ For choosing one condition out of two we use if-e and if there are more than two conditions we do nesting of if but if there are lot of conditions, nesting does not look readable, so switch control is next option.

→ The switch control statement is made up of three keywords. switch case default.

→ The general form of switch statement is as follows.

```
switch (expression)
{
    case value 1 ;
        Block-1 ;
        break ;
    case value 2 :
        Block-2;
        break ;
    case value 3 :
        Block-3;
        break;
    default:
        default block;
}
statement-x;
```

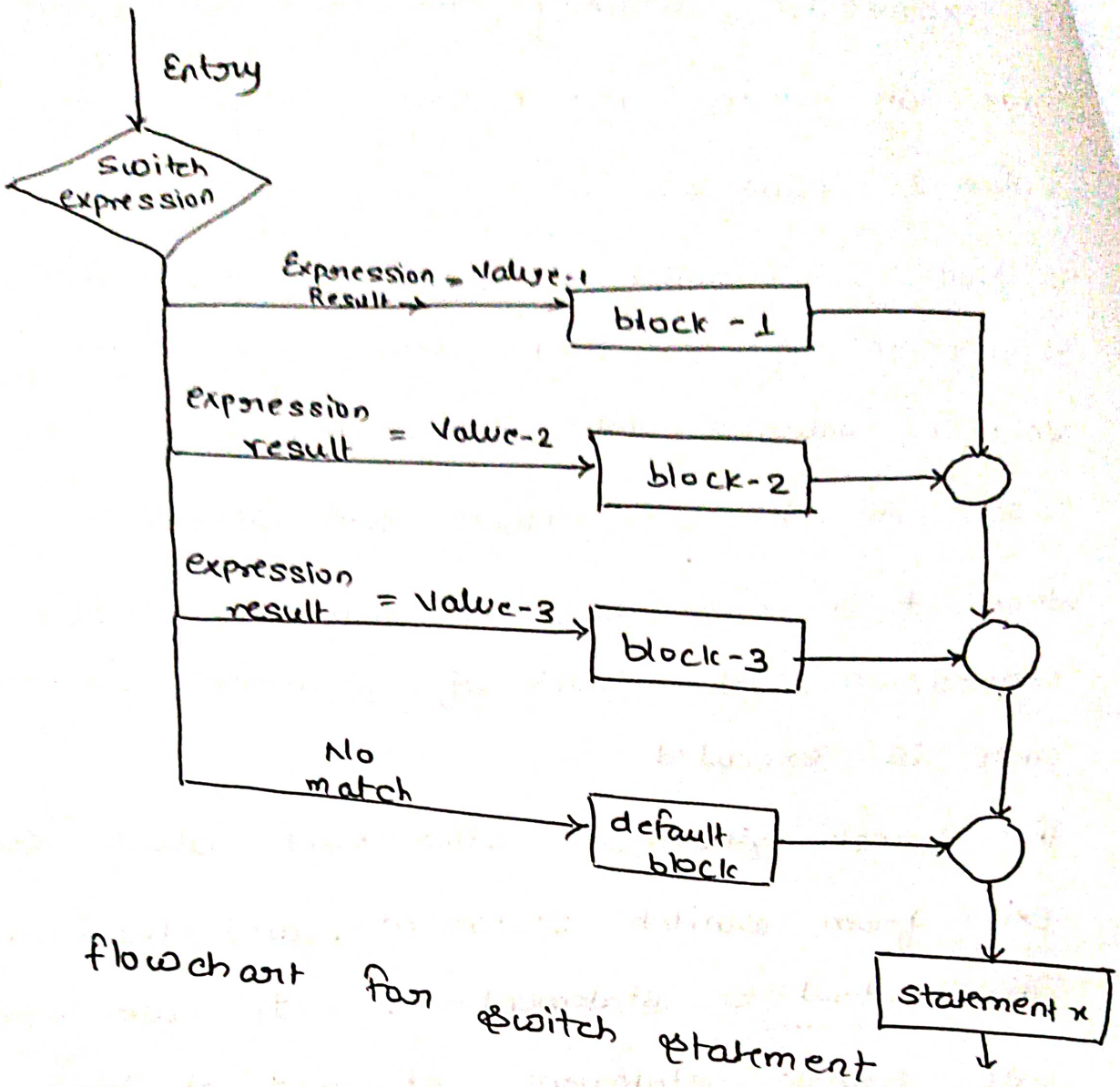
The expression following the keyword switch will have an integer or character value (value-1, value-2, value-3)

→ As soon as switch is executed, the value of expression is compared with case labels i.e. value-1, value-2, value-3 written after keyword case. All the case labels are unique.

→ When a case label is same as value of expression, the block of statement under that case is executed.

→ The break statement after each block causes exit from switch statement, and transferring the control to statement-x. If we will not use break statement at end of block, all the block statement after that block will be executed.

→ Default statement is executed if value of expression does not match with any of the case label. The default is optional case, if we don't use default in case of not matching any case label with value of expression control will go to statement-x.



flowchart for switch statement

→ if we use switch within another switch that is not known as nested switch statement.

→ We can use default anywhere in the switch body but we have to use break to come out of switch control.

→ For less no of cases if-else is faster but for large number of cases switch control works faster

GOTO statement

→ To branch unconditionally from one point to another in the program GoTo statement is used.

→ The general form of goto statement is shown below.

```
goto label;
- - - - - - -
- - - - - -
- - - - - -
label :
statement;
```

```
label :
statement;
- - - - -
- - - - - -
- - - - - -
goto label;
```

→ goto structure uses a label to indentify the point where control is to be transffered. label is a valid variable name followed by colon.

→ label is placed immidiately before the statement where control is to be transffered.

→ label can be before or after the program goto.

→ If label is before goto, loop will be formed and same statements will be executed repeatedly

→ If label is after goto or at last the program will run for ∞ loops.

algorithm flowchart

Iterative (Repeatative) Control instruction / Control instruction

→ If we want to repeat a set of instruction in program either a specified number of times or until a particular condition is satisfied, we use loop control instruction

→ Looping can be done using

i) The while statement
ii) The do-while statement
iii) The for statement

While statement

→ This is simplest of all looping instruction.

→ The general form of while is -

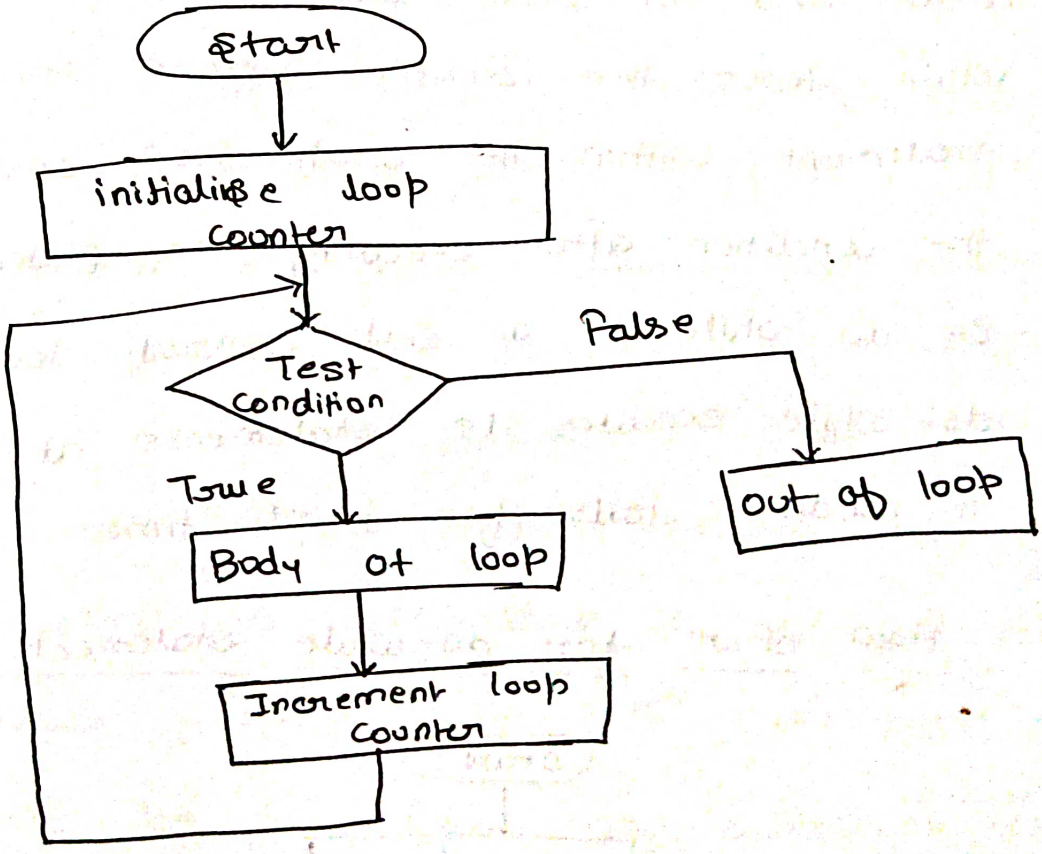
```
initialise loop counter ;
while (test loop counter using a condition)
{   statement 1 ;
    statement 2 ;
    increment / decrement loop counter ;
}
```

→ First loop counter which is a variable is initialised with some constant.. Then a test condition for loop counter is checked which is followed by while.

if While condition is true while statement is executed and loop counter is incremented / decremented and again condition is checked with for new value of loop counter.

→ This process continues until the result of loop counter become false

Flow chart for operation of while loop



→ While is an entry controlled loop statement as the control condition is tested before the start of loop execution.

The do-while statement

→ The do-while loop looks like this

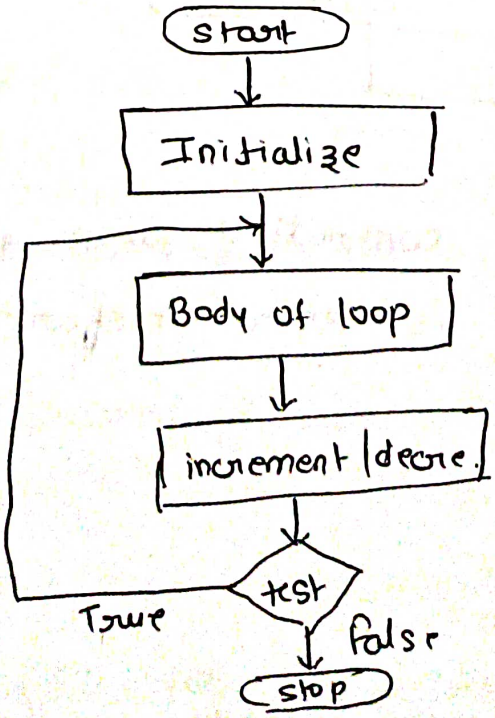
```
do
{   statement-1 ;
    statement 2 ;
    statement 3;
}   while ( test condition / loop counter )
```

→ There is a minor difference between the working of while and do while loop.

→ While tests the condition before executing any of the statement within the while loop and do-while tests the condition after executing the statements within loop so do while is an exit control loop statement.

→ do-while executes its statements at least once, even if condition fails for first time.

flow chart for do-while statement



...ical ... the algorithm.

The for statement

The for statement is entry controlled loop that provides a concise loop control structure.

→ The general form of for loop is -

```
for (initialization ; test condition ; increment )
{
    body of for loop
}
```

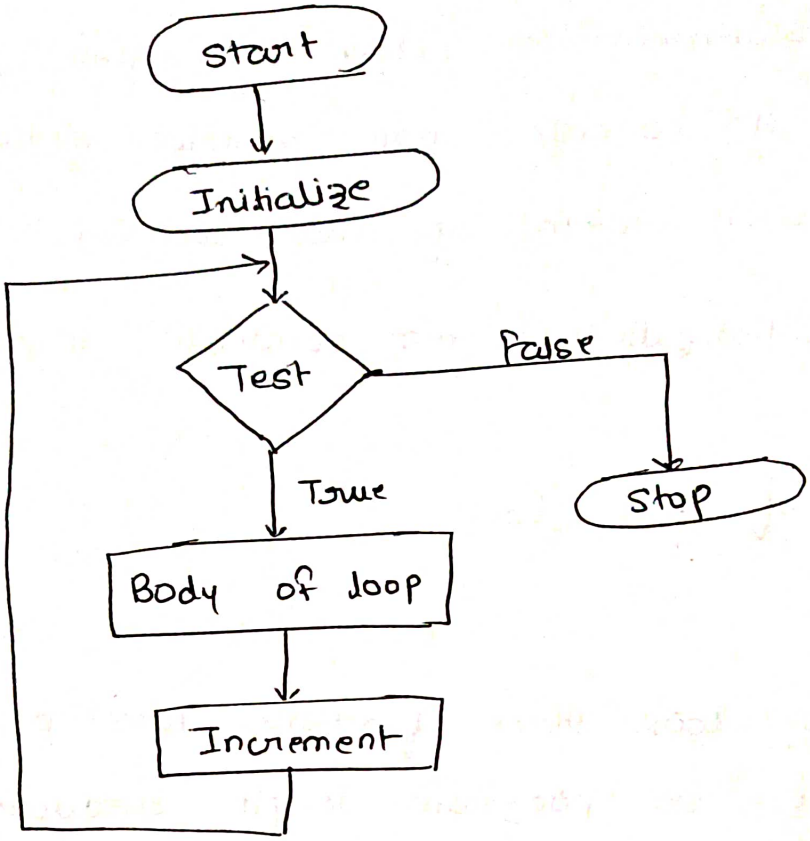
→ So in for loop three functions are performed in the same line, so program length reduces.

→ first initialization of ~~control~~ variable loop counter is done, then loop counter is tested for loop execution and finally if condition is true then loop statements are executed. Upon reaching closing brace of for, control is sent back to the for statement where value of loop counter is incremented / decremented by 1. Again loop counter is tested for loop statement.

→ if we don't want to initialize or increment in for loop, then we can write as

```
for ( ; test condition ; )
{
    for loop body
}
```

Flowchart of for loop



→ if we want to do multiple initialization, then we can write as:

for (initialization 1, initialization 2; test condition; increment)

→ similarly we can perform multiple increment operation also

The break statement

The keyword break can be used in the loop body or switch body.

→ The purpose of break is to terminate loop's execution immediately as it encounters & and passes the control to first statement after the loop.

→ break is always used with if, we use break in program where we want to break a Program in special condition.

→ General form of break is

```
break;
```

The continue statement

→ The keyword continue is used in the loop body.

→ The purpose of continue is to shift the control to the beginning of the loop skipping the statements of loop after it.

→ General form of continue is

```
continue;
```

→ Continue is used with if.

Algorithm :- A logical step by step

in different loop continue is executed as-

```
while (test condition)
{
    statement 1 ;
    if ( expression)
        continue ;
    statement 2 ;
}
```

```
do
{
    statement 1 ;
    if (expression)
        continue ;
    statement 2 ;
}
while (test condition)
```

~~for~~

```
for (initialization ; test condition ; increment)
{
    statement 1 ;
    if ( expression)
        continue ;
    statement 2 ;
}
```