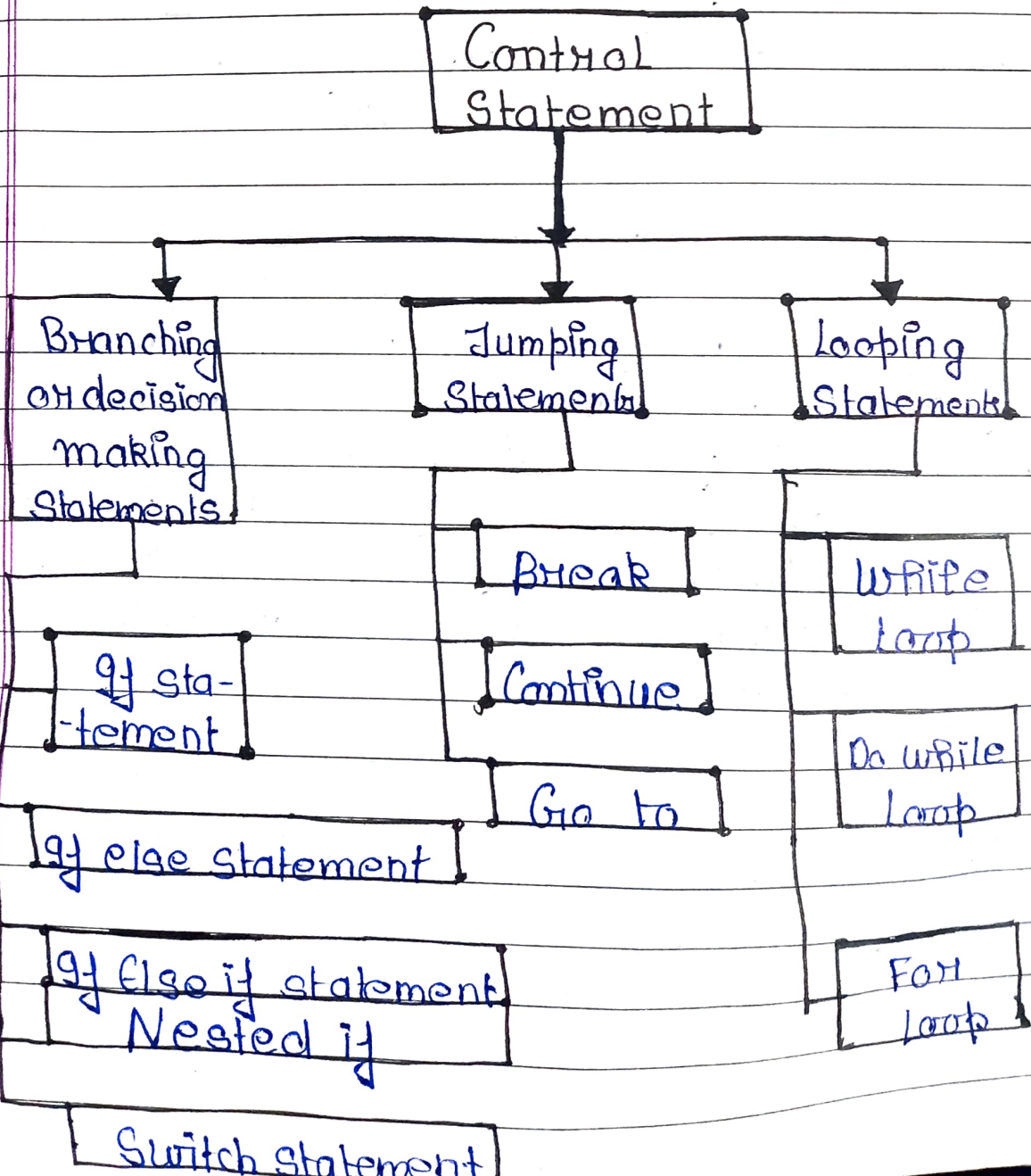
Chapter - 5
Control Statements

Control Statements :- Control statements are the statements which are used to control the flow of execution of the instructions written in a program. Simply the instructions are executed sequentially. This type of control structure is known as sequence control structure.



1. Branching :- Branching is a procedure of executing a block of statements on the basis of the result of Condition. This is done through decision making statement

2. Jumping :- Jumping is a procedure of transferring execution control to a particular point in program. This is done through jumping statements.

3. Looping :- Looping is a process of repeating a set of statements until a condition hold true. Three types of looping statements are used to accomplish this task.

Decision making Statements :- Or Selection Constructs are used for Accomplishing branching in C programs. These Statements select one of the two or more execution paths, depending on the result of Condition. This means, if the Condition is true, one set of statement is executed and if the Condition is false, other set of statements is executed. C provides following decision making statements

1. if statement
2. if else Statement
3. if else if statement
4. Nested if
5. Switch statement

1. if Statem

to execu
Condition
the Gen

Syntax :-

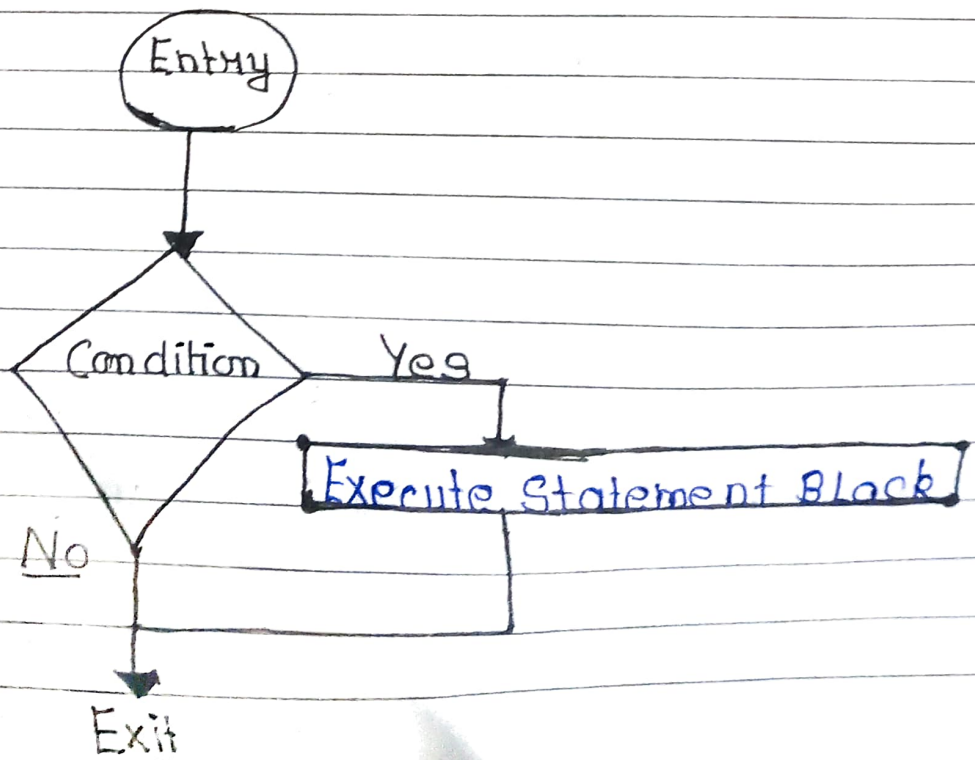
if (condi
{
Statement
Statemen
- - - -
}
Statemen
Supplied
while b
with C
Illustra

4.1 If Statement :- If statement is a Conditional or Selection Construct that is used to execute a set of statements, if Certain Condition is true
The General Syntax of if Statement is :-

Syntax :-

```
if (condition)
{
Statement 1;
Statement 2;
- - - - - - -
}
```

Statement 1 and 2 will be executed only if the Supplied Condition results to true. Otherwise the whole block will be skipped and the execution will continue after if block. Below Flowchart illustrates the use of Simple if statement



Example :-
To Check whether a student is pass in the examination

```
// Program to check whether a student is pass
# include <stdio.h>
# include <conio.h>
Void main()
 {
int marks;
clrscr();
Printf("\n Enter your marks in exam:");
Scanf("%d", &marks);
if(marks>=35)
{
 printf("You are pass");
 printf("Congratulation");
}
getch();
}
```

Illustration :- Program inputs value of marks and stores in variable marks and displays the message of Pass and Congratulations, if the entered marks are greater than or equal to 35.

[If there is only one statement under if block then, there is no need to put Curly braces {} they are required only when the number of statements is more than one]

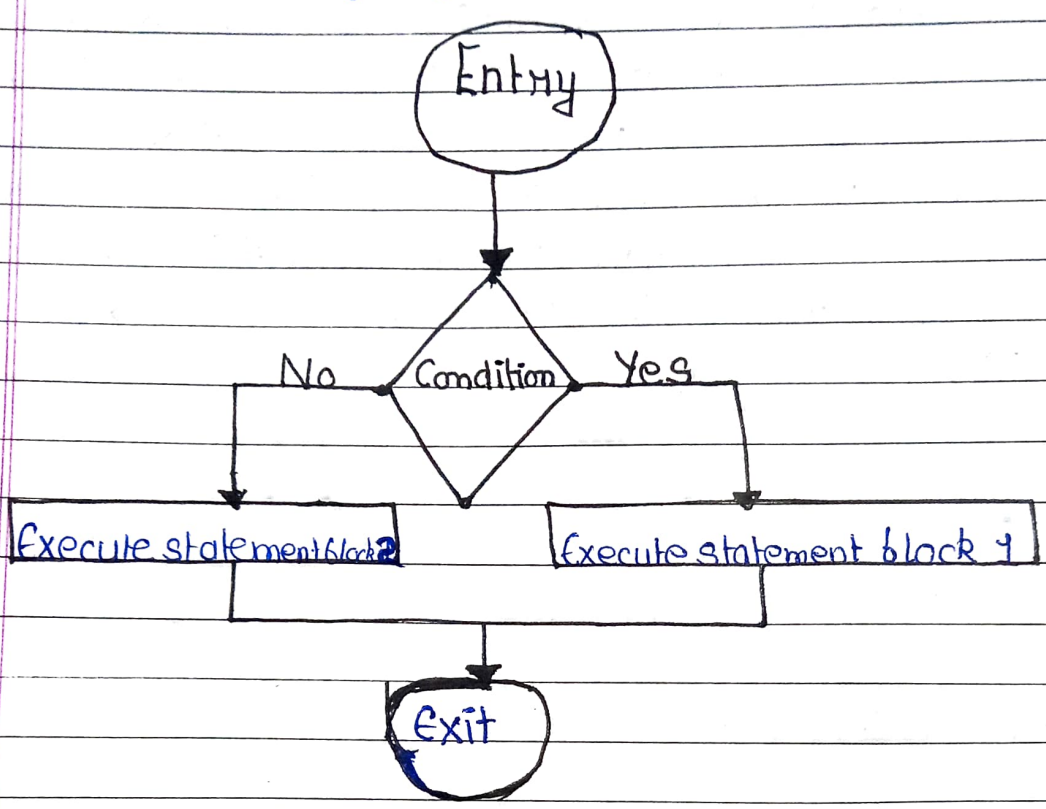
2.4 If Else Statement :- If else Construct is also a Conditional Construct of c language. It is used to execute a set of statements, if the condition is true and some other set of statements, if the condition is false. If else construct is an extension of if statement, it includes a condition a statement block to be executed when condition is true and a statement block to be executed, if the condition is false

Syntax :-

```
if (condition)
{
Statement block 1
- - - - - -
}
else
{
Statement block 2
- - - - -
}
```

If the condition is true then statements in the first block will be executed and if the condition is false, then the statements of the second block will be executed. if else statement is used in the situations when the conditional statement can only have two possible results, one execution path is followed when the first result is obtained and other execution path is followed.

When the second result is obtained. below Flowchart of if else logic program illustrates the working of if else Construct.



Flowchart For if else logic

Example Program:

Write a program to input a number from User and print whether the number is even or odd.

```
//Program to check whether a number is even or odd.
#include <stdio.h>
#include<conio.h>
Void main()
{
int num;
printf("\n Enter a number");
Scanf("%d",&num);
```

```
if(num%2 == 0)
{
 printf("\n You entered even number");
 }
 else
 {
 Printf("\n You entered odd number");
 }
}
```

Illustration :-

Program inputs a number and stores it in a variable "num". If the value of num is further divisible by 2, the first block of statements is executed, otherwise the second block executed.

3. Nested if statements :- When an if statement is written inside the body of other if statement. It is called nested if statement. In case of nested if statements, the inner if statement is executed only if the outer evaluates to true. Below program illustrates the use of nested if statement

```
//Program to check whether a number is divisible
 by 2 and 3
#include <stdio.h>
#include<conio.h>
 Void main()
 {
  int num;
```

```
printf("\n enter a number");
scanf("%d", &num);
if(num%2==0)
{
    printf("\n entered number is divisible by 2");
    if(num%3==0)
    {
        printf("\n entered no is divisible by 3");
    }
}
getch();
}
```

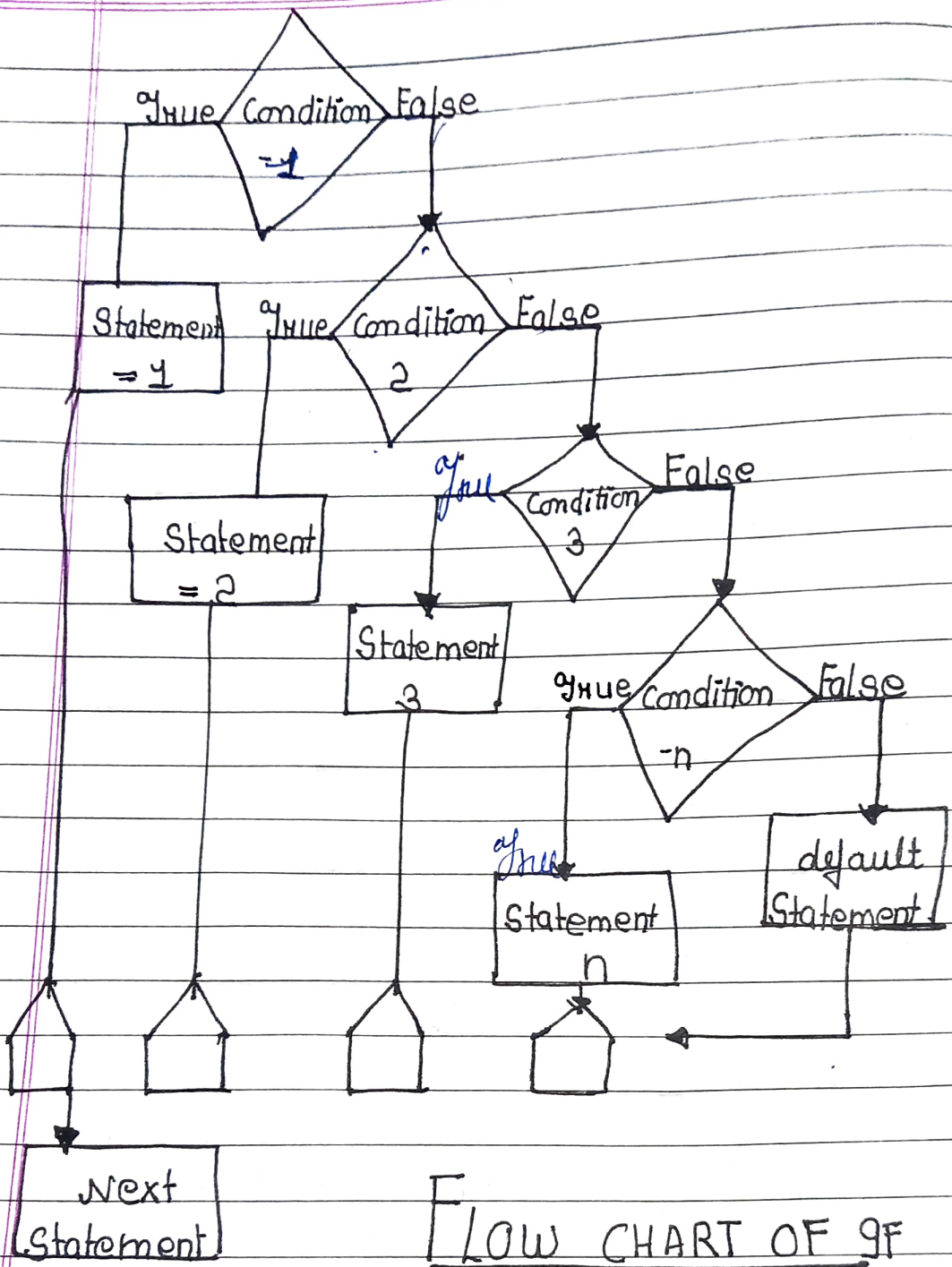
Illustration:-
The program inputs a number and stores it in a variable "num". If the number is completely divisible by 2, a message is printed and then nested if statement is checked. If the number is also divisible by 3, another message is printed.

4.) if else if statement :- if else construct works for situations where there are two possible execution paths for a particular condition. In practical problems, there exist multiple execution paths for a condition. For such kind of problems, C has a support for if else if construct. This construct works with the conditions having multiple possible outcomes, if else if construct is used when the condition has multiple possible outcomes.

Syntax :-

```
if (condition 1)
{
Statement block 1
---
}
else if (condition 2)
{
statement block 2
---
}
else if (condition 3)
{
Statement block 3
---
}
```

while executing if else if construct, c compiler checks the conditions from top to bottom. If first condition true then statement block 1 would be executed and rest of conditions will not be checked. That means all steps till the end will be skipped and no other condition will be check. If condition 1 is false, then only Condition 2 will checked. If condition 2 is true then statement block 2 will be executed and rest all will be skipped. If condition 2 is also false, then only Condition 3 will be checked and statement block will be executed below Flowchart illustrates the implementation of if else if construct



FLOW CHART OF IF ELSE if Logic

Program to demonstrate if else if Construct:

```
# include <stdio.h>
# include <conio.h>
Void main()
{
```

```
int mks;
printf("\n Enter your marks");
Scanf("%d",&mks);
if(mks<35)
{
printf("\n I am sorry! You are FAIL");
}
else if((mks>=35)&&(mks<45))
{
printf("\n You have got c grade");
}
else if((mks>=45)&&(mks<60))
{
Pofnf("\n You have got B grade");
}
else if (mks>60)
{
printf("\n You have A grade");
}
getch();
}
```

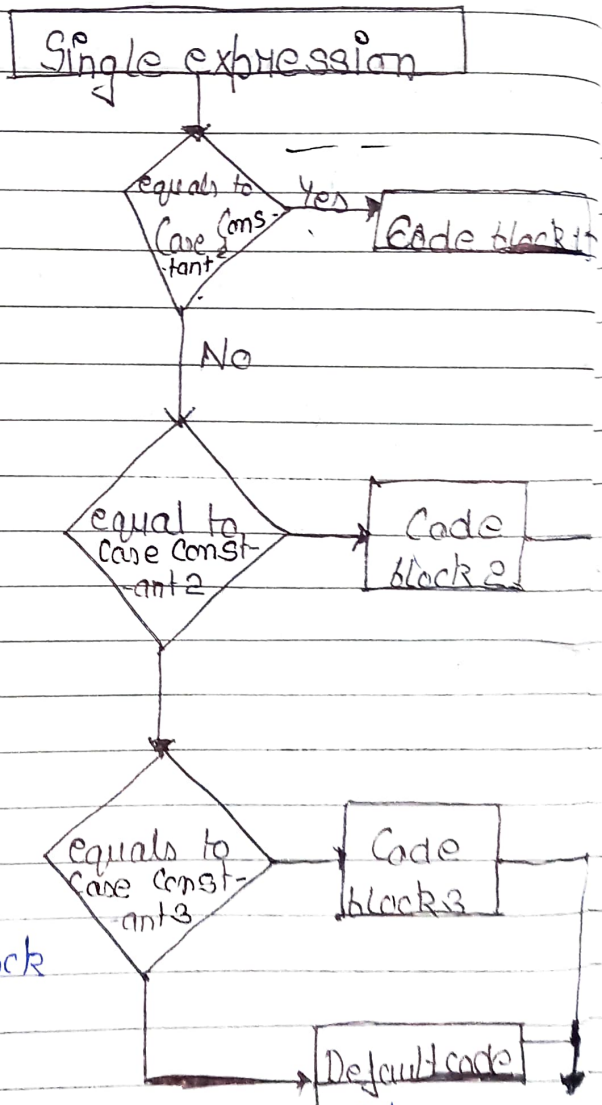
Output:-

Enter Your marks: 58
You have got B grade

5.4 Switch Case Construct :- When the number of choices after a condition are large, switch case construct is used. Switch Case Construct is a multiway decision construct like if else if construct and performs the same functionality.

Syntax :-

```
Switch (variable)
{
case value 1:
statement block 1
break;
Case value 2:
statement block 2
break;
.....
.....
Case value n:
statement block n
break;
default:
default statement block
break;
}
```



In Switch Case construct, the variable on which the switch has been applied can be a valid integer or character variable

Program to check whether Entered character is Vowel or Consonent :-

```
#include<stdio.h>
#include<conio.h>
Void main()
{
Char ch=0;
clrscr();
printf("Enter a character :");
Scanf("%c",&ch);
Switch(ch)
{
case 'a':
printf("Entered character is Vowel \n");
break;
Case 'e':
printf("Entered character is vowel");
break;
Case 'i':
printf("Entered Character is Vowel");
break;
case 'o':
printf("Entered character is Vowel");
break;
Case 'u':
printf("Entered character is Vowel");
break;
default:
printf("Entered character is Consonent");
}
```

getch();
}

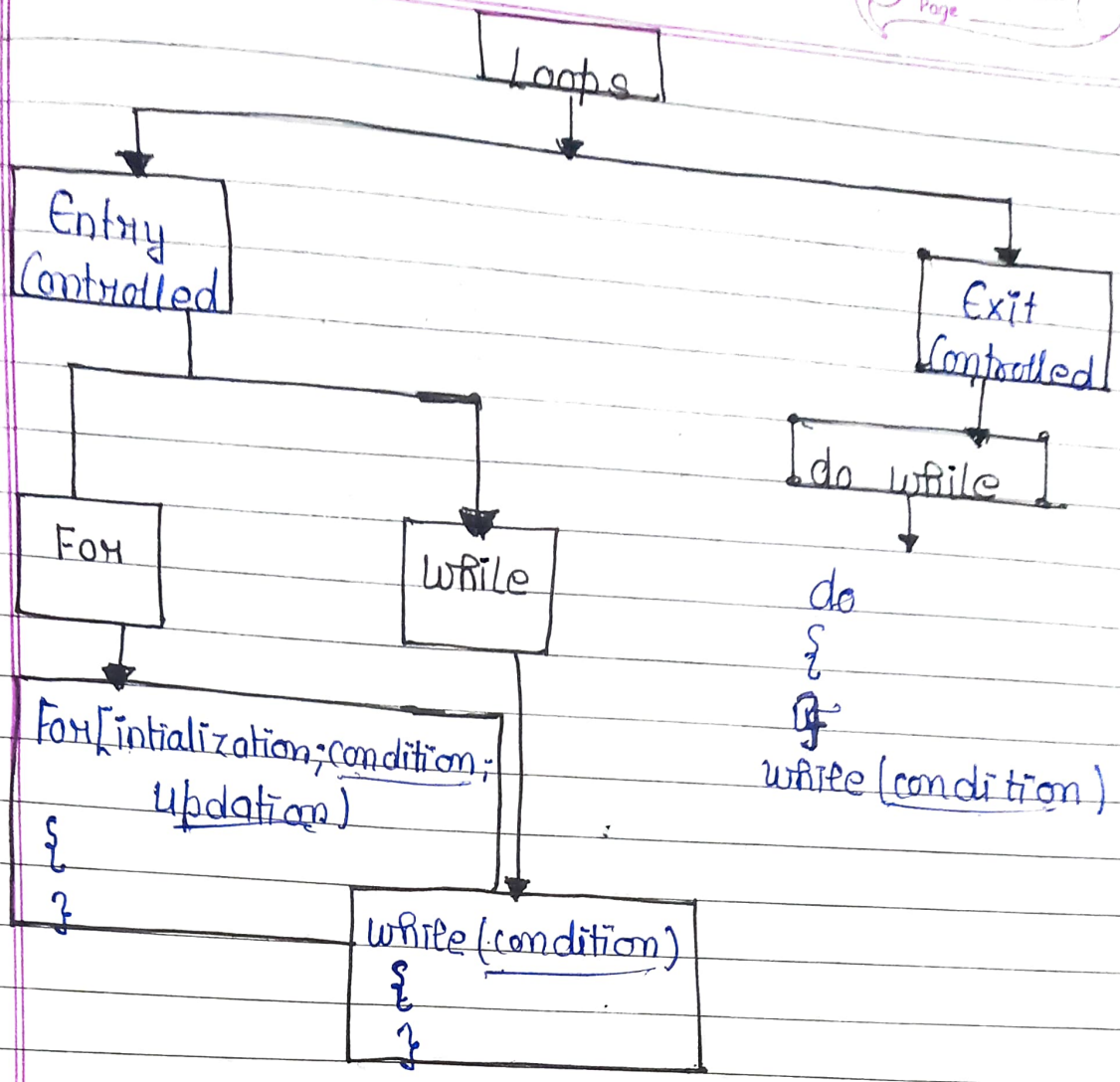
2.4 Iteration or Looping Constructs :- The Concept of repeating the execution of a particular block of statement till a conditional expression is satisfied is called as Iteration or looping. Looping Constructs reduce size of the program files and iterations from programmer's mind.

In loop, the statement needs to be written only once and loop will be executed 10 times as shown below. In Computer programming, a loop is a sequence of instructions that is repeated until a certain Condition is reached

Types of Loops :- There are mainly two types of Loops :-

1.4 Entry Controlled Loops :- In this type of loops the test Condition is tested before entering the loop body for loop and while loop are Entry Controlled Loops.

2.4 Exit Controlled Loops :- In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore the loop body will execute atleast once, irrespective of whether the test condition is true or false do-while loop is exit Controlled Loop.



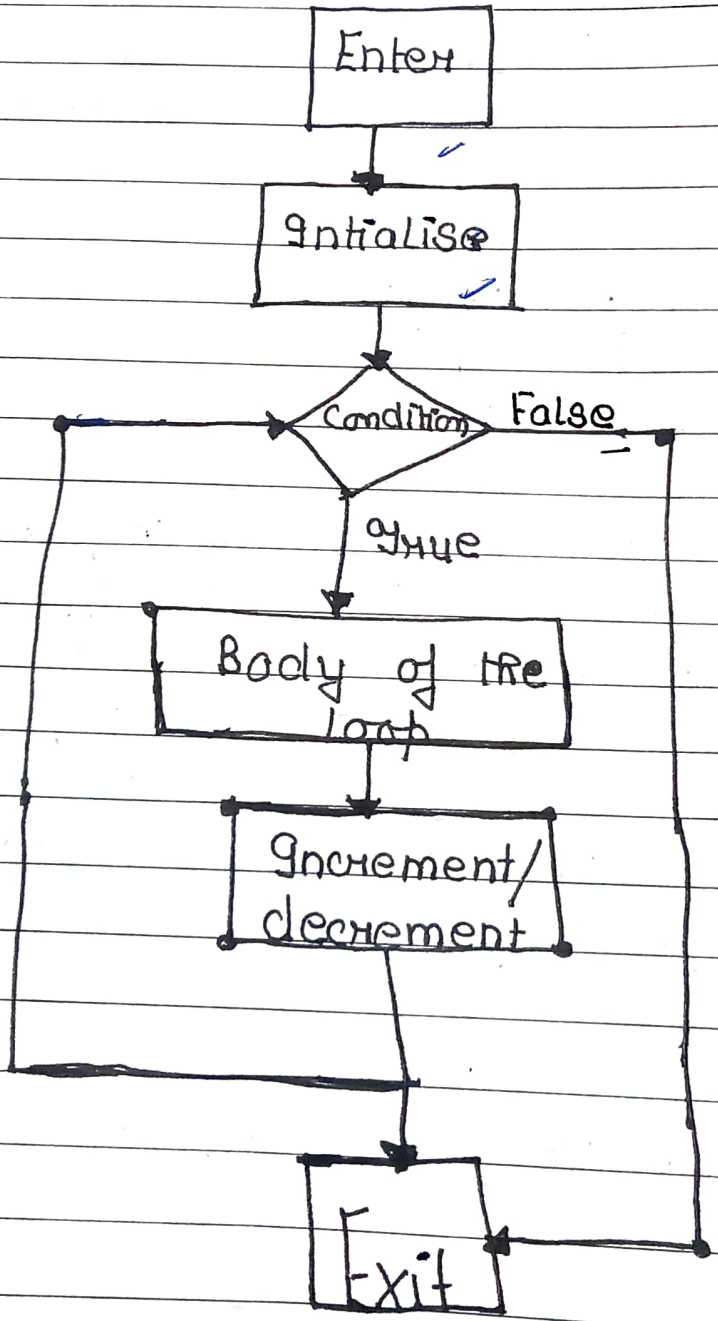
4. For Loop :- For loop is the most poputar looping Construct. It is very different from while and do while loops. The main advantage of for loop is that it contains all the three phases of loop in single step. The Intialize, condition and Increment phase are contained in single statement. This makes the syntax of this loop a bit simpler than the others. There is no need of Intializa of Conditional vasiable inside or outside the loop body. The Syntax of for loop is :-

Syntax :-

For (intialization phase; condition phase; Increment decrement ph

{
loop body
}

Initialization phase generally consists of an assign-ment, condition phase contains expression that will be tested to carry out looping. The increment or decrement phase increments or decrements the value of the conditional variable. The initia-lization phase is carried out only once in for loop



— Flow chart for Working of for loop —

Program to Illustrate the working of for Loop

```
// Program to print hello world 10 times by using for loop
#include <stdio.h>
#include<conio.h>
Void main()
{
int i=0;
for(i=1; i<=10; i++)
{
printf("Helloworld \n");
}
getch();
}
```

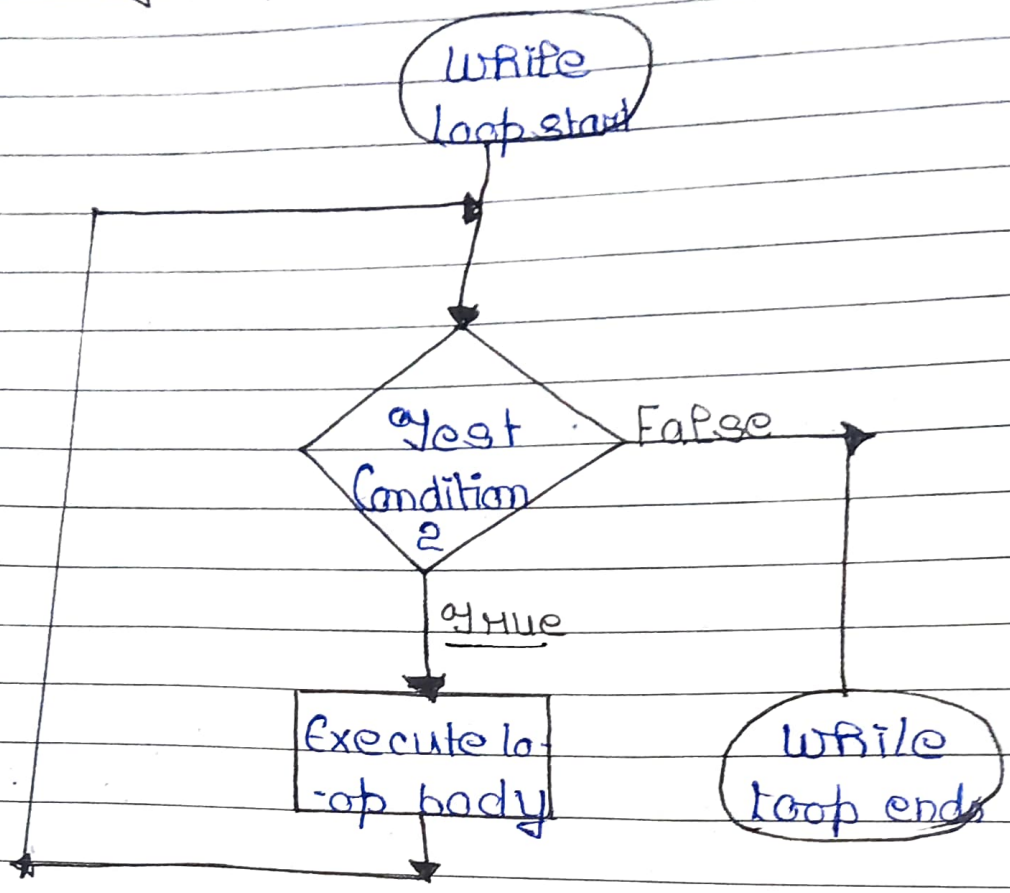
2. While Loop :- While studing for loop we have seen that number of [times the] iterations is known beforehand. the number of times the loop body is needed to be executed is known to us. while loops are used in situations where we don't know the exact number of iterations of loop beforehand. The loop execution is terminated on basis of test condition.

Syntax :-

```
Intialization expression;
while(test_expression)
{
  // statements
  update_expression;
```

}

Flow diagram/chart :-



Example program :-

```
//Program to print 'hello world' using while loop
#include <stdio.h>
#include<conio.h>
Void main()
{
   int i=1;
   while(i<6).
   {
      printf("Hello world \n");
      i++;
```

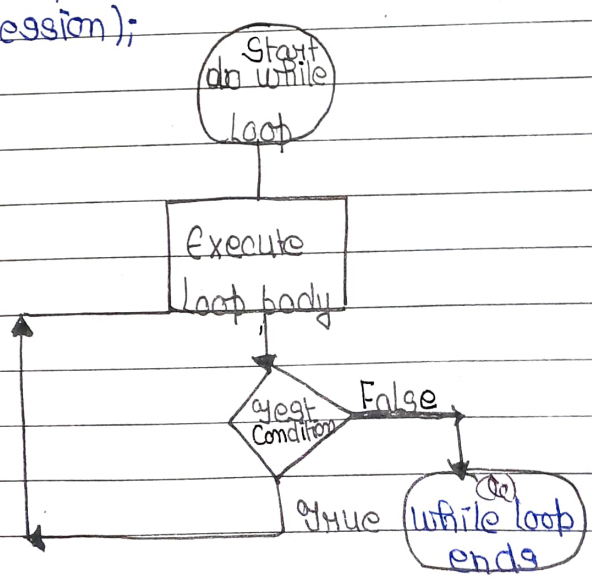
}
getch();
}

3.4 Do while Loop :- In do while loops also the loop execution is terminated on the basis of test condition. The main difference between do while loop and while loop is in do while loop the condition is tested at the end of loop body ie do while loop is exit controlled whereas the other two loops are entry controlled loops. In do while loop the loop body will execute at least once irrespective of test condition. do while loop is post tested looping construct in which body of loop is executed first and then condition is tested.

Syntax :-

```
intialization expression;
do
{
  // statements
  update_expression;
} while (test_expression);
```

Flow chart :-



Example :-

```
//c program to illustrate do while loop
#include <stdio.h>
#include <conio.h>
Void main()
{
    int i=2;
    do
    {
        //Loop body
        Printf("Hello world \n");
        i++;
    }while(i<1);
    getch();
}
```

Illustration :-

In above program test condition (i<1) evaluates to false. But still as the loop is exit controlled the loop body will execute once

Infinite loop :- An infinite loop is a piece of coding that lacks a functional exit so that it repeats indefinitely. An infinite loop occurs when a condition always evaluates to true. Usually this is an error.

3.7 Jumping Statement :- With the use of Loop, the execution of certain statements can be repeated for a number of times but we cannot alter the execution sequence of program ie the order in which the different statements of the program are executed. In standard way, the execution of a program starts from the first statement in body of main() function and continues downwards. The conditional constructs can be used to branch the execution on particular path. C Language offers a way of altering the sequence of execution. This is with the help of jumping statements

Types of Jumping statements :-

1.) goto statement
2.) break statement
3.) Continue statement

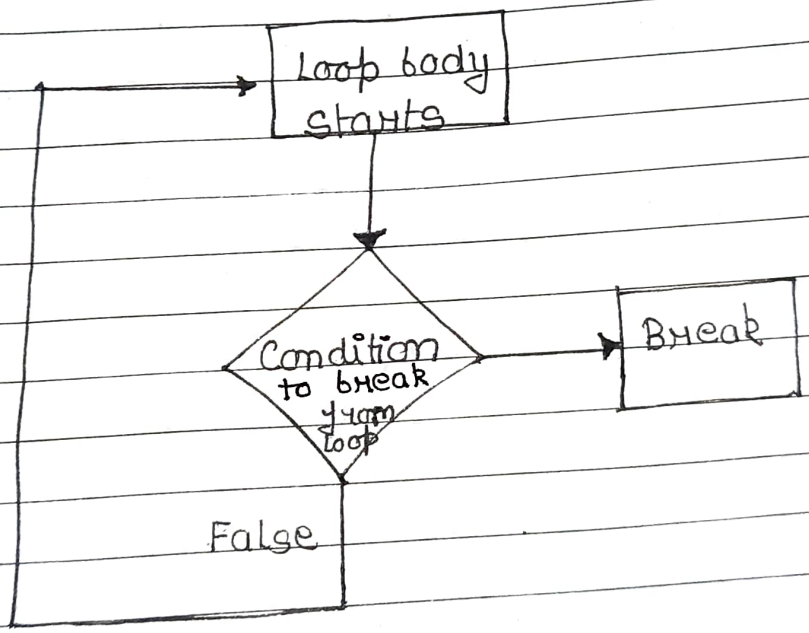
1.) Break statement :- The Break statement is used to terminate the loops. The termination of the loop can occur in two ways: normal termination and abnormal termination. The normal termination of Loop occurs when the condition phase of the loop results in false. In some cases we need to break the execution of ~~loop.~~ [~~Break~~] depending on the value input by User. This is called Abnormal termination of loop. break statement is used in

Such cases. whenever the break statement is used, the control comes to first statement after loop body. The use of break statement reduces the number of unnecessary executions of loop. This way, it speeds up the program, executions. It is used with decision making statement. It forces the loop to stop execution of further iteration.

Syntax:-

```
break;
```

Flow chart:-



Program:-

// Program to print numbers from 1 to 5 and skipping 5 number

```
#include<stdio.h>
#include<conio.h>
Void main()
{
  int i=0;
```

```
clrscr();
For (i=1; i<=10; i++)
{
if (i==5)
break;
printf("%d", &i);
}
getch();
}
```

2. Continue Statement :- In some specific programming problems, we need to skip the remaining statements the loop body and take the Control to the beginning of the loop. In situation like this. Continue statement is used Continue statement is used to execute other parts of loop while skipping some parts declared inside condition, rather than terminating loop. It Continues to execute the next iteration of same loop. It is used with decision making statement which must be present inside loop. This statement can be used inside for, while or do while loop

Syntax :-

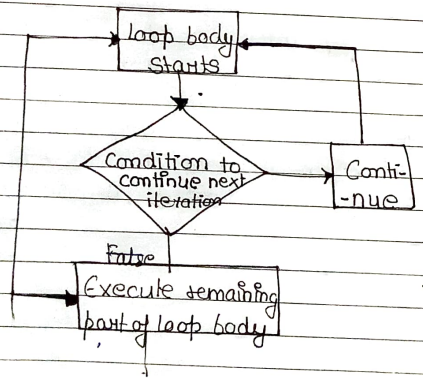
```
Continue;
```

Example :-

```
//Program to print number from 1 to 10 and skipping
5 from 1 to 10 Sequence
#include<stdio.h>
```

```
#include<conio.h>
Void main()
{
int i=0;
clrscr();
for(i=1;i<=10;i++)
{
if(i==5)
Continue;
printf("%d",i);
}
getch();
}
```

Flow chart :-



3.4 Goto statement :- The goto statement is used to alter the sequence of a C program and start execution from a desired statement. Whenever we use goto statement with a label, the control jumps backward or forward to search the label and executes statements onwards.

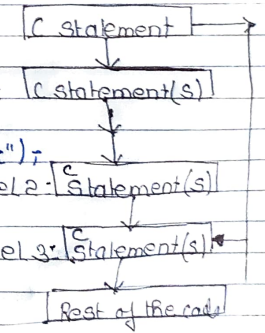
Syntax :-

```
goto label
Statements
Statements
label: statement
Statement
```

Example :-

```
//Program to check number is even or odd using goto statement
#include<stdio.h>
#include<conio.h>
Void main()
{
int i;
clrscr();
printf("Enter the value of a :");
scanf("%d",&i);
if(i%2==0)
goto label1;
else
goto label2;
label1:
printf("\n Number is even");
```

Flow chart :- [Goto statement]



```
Label 2:
puts("\n number is odd");
getch();
}
```

Nested if flow chart :-