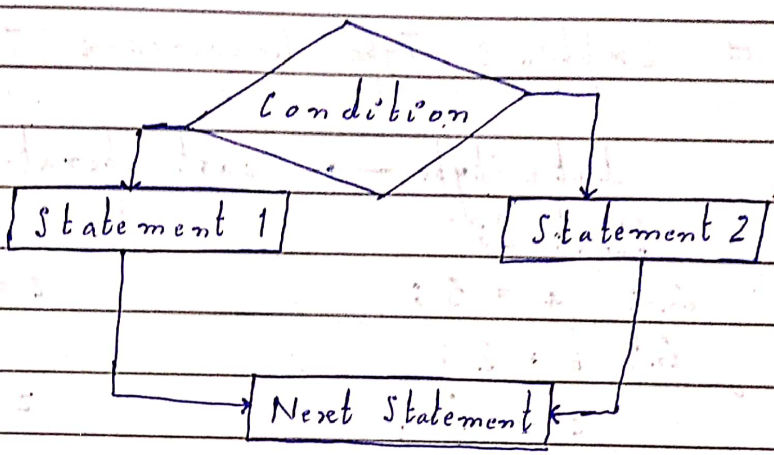
```
if (condition)
    statement 1;
else
    statement 2;
```




```
if (condition)
{
    statement;
    =
}
else
{
    statement;
    =
}
```

→ Program to print a message if negative number is entered.

```
#include <stdio.h>
main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num < 0)
    printf("Number entered is negative \n")
    printf("The entered number is %d", num);
}
```

→ Program to print the larger and smaller of two numbers.

```
#include <stdio.h>
main()
{
int num;
printf("Enter a number: ");
scanf("%d", &num);
if (num < 0)
printf("
```

```
{ int a,b;
printf("Enter the first number: ");
scanf("%d", &a);
printf("Enter the second number: ");
scanf("%d", &b);

if (a > b)
printf("larger no = %d and smaller no = %d \n", a, b);
else
printf("larger no = %d and smaller no = %d", b, a);
}
```

→ Program to print whether the no is even or odd.

```
#include <stdio.h>
int main(void)
{
    int num;
    printf("Enter no: ");
    scanf("%d", &num);

    if (num%2 == 0)
    printf("Number is even");
    else
    printf("Number is odd");
    }
```

→ Nesting of if ... else

We can have another if ... else statement in the if block or else block. This is called nesting of if ... else.

```
if (cond 1)
{
    if (cond 2)
    stmt A1;
    else
    stmt A2;
}

else
{
    if (cond 3)
    stmt B1;
    else
    stmt B2; }
```

→ Program to find largest number from 3 given number

```
#include <stdio.h>
int main(void)
{
    int a, b, c, large;
    printf("Enter 3 no: ");
    scanf("%d %d %d", &a, &b, &c);

    if (a > b)
    {
        if (a > c)
        large = a;

        else
        large = c;
    }

    else
    {
        if (b > c)
        large = b;

        else
        large = c;
    }

    printf("Largest no is %d", large);
}
```

→ else if ladder

Type of nesting in which there is an if ... else statement in every else part except the last else part.

```
if (cond 1)
    stmt A;

else if (cond 2)
    stmt B;

else if (cond 3)
    stmt C;

else
    stmt D;
```

→ Program to find out the grade of the student when the marks of 4 subjects are given; The method of assigning grade is given as

per >= 85, grade = A
per < 85 and per >= 70, grade = B
per < 70 and per >= 55, grade = C
per < 55 and per >= 40, grade = D
per < 40 and , grade = E

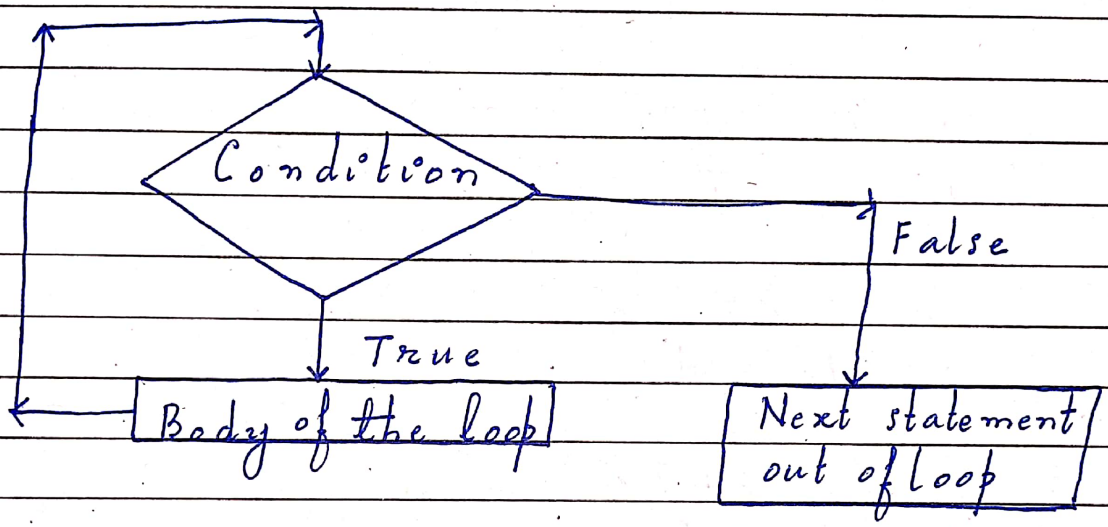
```
#include <stdio.h>
int main(void)
```

```
{
    float m1, m2, m3, m4, total, per;
    char grade;
    printf("Enter marks for 4 digits");
    scanf("%f %f %f %f", &m1, &m2, &m3, &m4);

    Total = m1 + m2 + m3 + m4;

    per = total / 4;

    if (per >= 85)
        grade = 'A';
else if (per >= 70)
        grade = 'B';
else if (per >= 55)
        grade = 'C';
else if (per >= 40)
        grade = "D";
else
        grade = 'E';
ff ("Per is %f and grade is %c", per, grade);
}
```

Loops

1. While
2. Do while
3. For

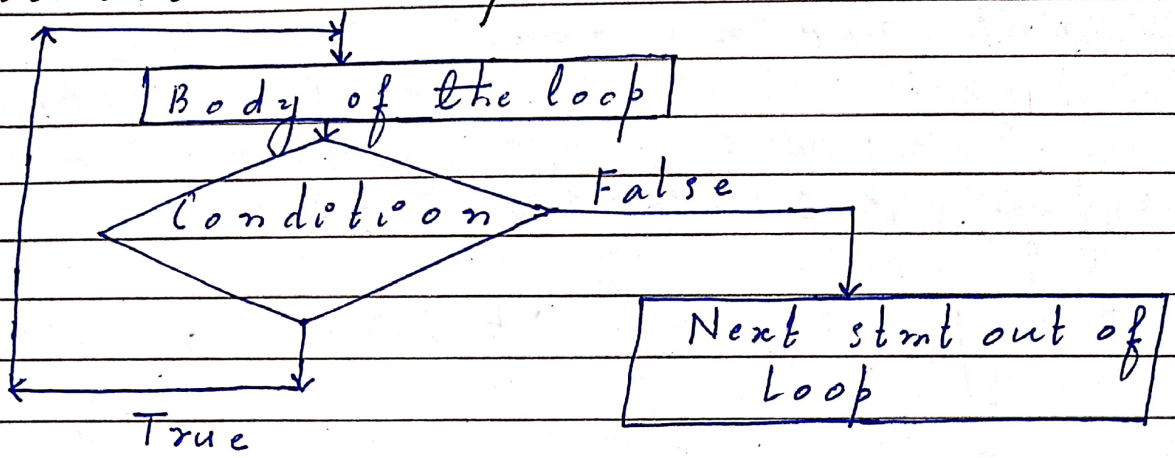
While Loop



WAP to print the numbers from 1 to 10.

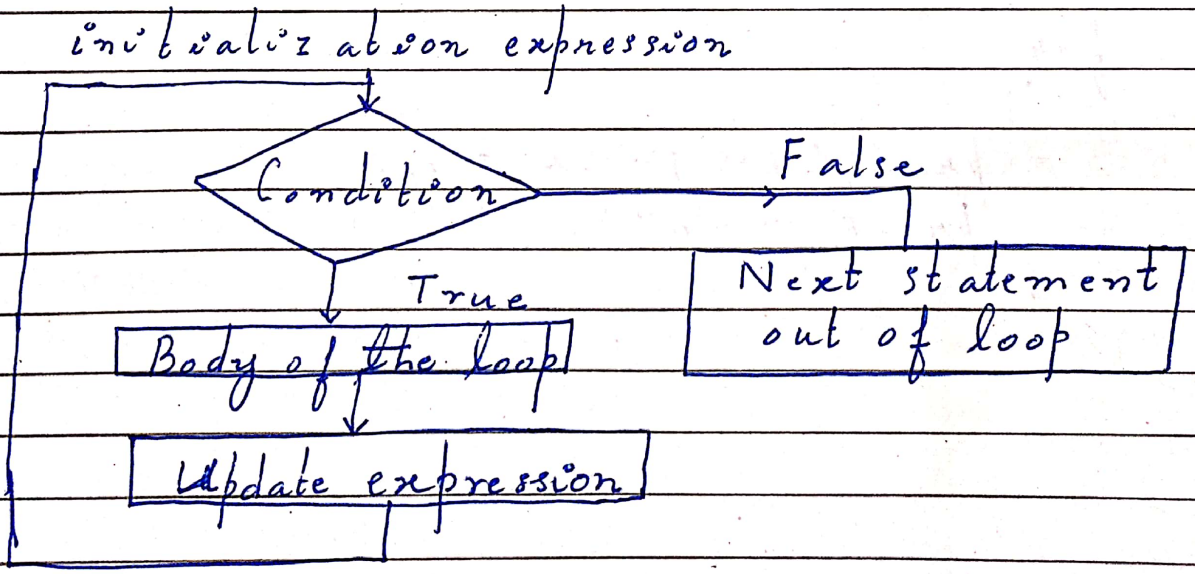
```
#include <stdio.h>
int main(void)
{
    int i=1;
while (i<=10)
{printf("%d\t", i);
    i= i+1; }
}
```

→ Do ... while Loop




```
#include <stdio.h>
int main (void)
{
    int i = 1;
do {
        printf("%d\t", i);
            i = i + 1;
        }
while (i<=10);
}
```

→ For Loop ;



```
# include <stdio.h>
int main(void)
{
    int i;
for (i=1; i<=10, i++)
  {
    printf ("%d \t", i);
  }
}
```

Nesting of Loop

When a loop is used inside the body of another loop.

```
# include <stdio.h>
{ int i, j;
  for (i=1, i<=3, i++)
    {
      printf ("i= %d \n", i);
    }

  for
    {
      printf ("j= %d \t", j);
      print ("\n");
    }
}
```

→ Break Statement

```
#include <stdio.h>
int main(void)
{
    int n;
for (n=1; n<=5; n++)
    {
        if (n==3)
    {
        printf("I understand the use of break");
        break;
    }
    printf("Num=%d", n);
    }
    printf("out of loop");
}
```

→ Continue Statement (Control Statement)

The continue statement is used when we want to go to the next iteration of the loop after skipping some statement of the loop.

```
#include <stdio.h>
int main(void)
{
    int n;
for (n=1; n<=5, n++)
{
    if (n==3)
```

```
{
    printf ("I understand the use of continue
             statement")
    continue;
}
    printf ("No = %d\n", n);
}
printf ("Out of loop");
}
```

In while and do while loops after continue statement the control is transferred to the test condition and then the loop continues, whereas in the for loop after continue statement the control is transferred to update expression and then the condition is tested.

Writing a program to check whether given number is prime or not.

```
#include <stdio.h>
#include <math.h>

int main()
{
    int i, num, flag = 1;
    printf ("Enter a number: ");
    scanf ("%d", &num);

    for (i = 2, i <= sqrt(num), i++)
{ if (num % i == 0)
    printf ("%d is not prime\n", num);
     flag = 0;
  break;
}
```

```
if (flag = 1)
{
    printf ("%d is prime\n", num);
}
```

→ Writing a program to calculate sum and average of 10 positive numbers.

```
#include <stdio.h>
int main()
{
    int i=1, n, sum=0;
    float avg;
    printf("Enter 10 positive numbers \n");
    while (i<=10)
    {
        printf ("Enter no %d : ", i);
        scanf ("%d", &n);
        if (n<0)
        {
            printf ("Enter only positive numbers \n");
            continue;
        }
        sum+=n;
        i++;
    }
    avg = sum/10.0;
printf ("Sum = %d and Avg = %f \n", sum, avg);
```

Switch

This is a multi-directional condition control statement.
can be any C expression yielding integer

```
Switch (expression)
{
    case constant 1:
     statement;
     ......

    case constant 2:
     statement;
     ......
     :
    case constant n:
     statement;
     ......

     default:
      statement;
}
```

Expression: It can be a value of an integer or character variable or a function called returning an integer, or an arithmetic logical or relational, bitwise expression yielding an integer.

Valid

```
int a, b, c
switch (a)
switch (d+e-3)
switch ( func (a,b))
```

```
switch (a > b)
switch (a > b && b > c)
```

Invalid

```
switch(f)
```

```
switch(a+4.5)
```

The constants following the case should be of integer or character type.

They can be either constant or constant expression.

~~float or strong constant~~

~~multiple constant in single case~~

Valid

```
case 4:
case 2+4:
```

```
case 'a':
case 'a' > 'b':
```

Invalid

```
case "second":
case 2.3:
case a:
```

```
case a>b:
case a+2:
case 2,4,5:
```

```
#include <stdio.h>
int main()
{
    int choice;
    printf("Enter your choice\n");
    scanf("%d", &choice);

    switch(choice)
    {
    case 1:
    printf("first\n");
     break;

    case 2:
    printf("second\n");
     break;
```