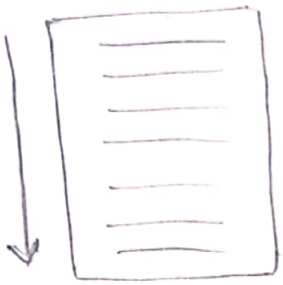
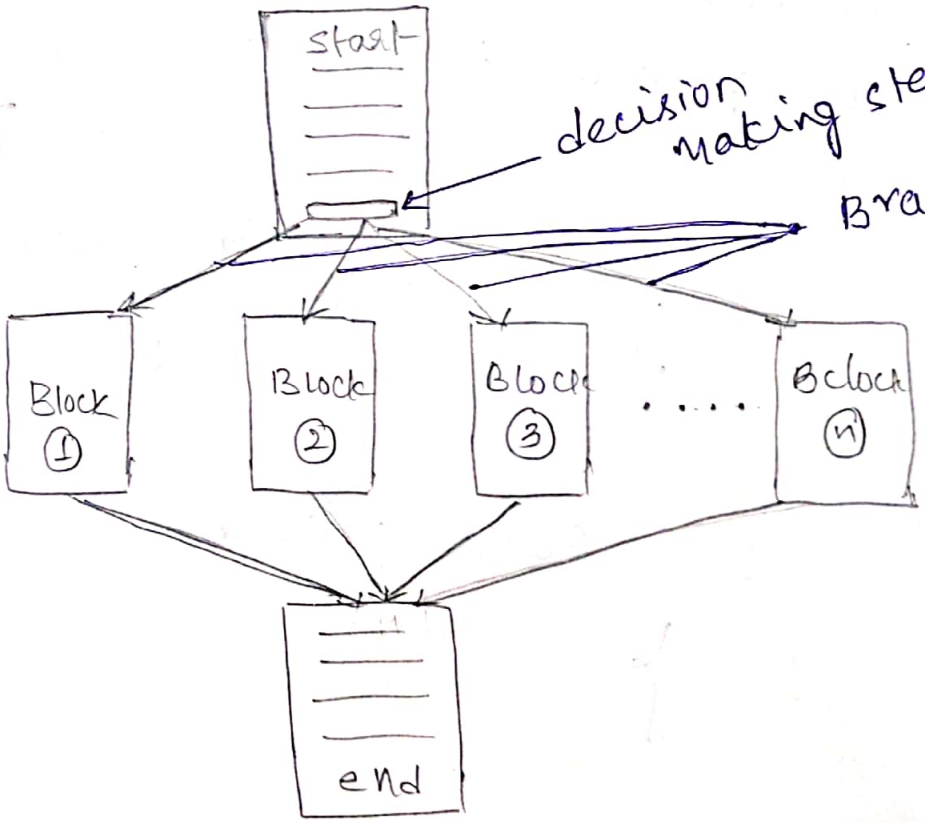
Decision Making & Branching

In C language, programs generaly run from top to down order in sequence.



In certain cases we need to make decision and execute certain block of code baled on the condition is called decision making & statement which makes decision is called decision making statement.



The multiple paths that the program takes are called branching (branches). The control may execute any one block based on the outcome of the condition. (True or False)

C provides following decision making statements.

- if
- if-else
- else if ladder
- nested if
- switch

① if statement :- It is used to execute a block of code if the condition is true.
[It handles only one condition]

if (Condition) → Relational expression or Logical expression.
{

Syntax

Block of code

}

Example :→ write a c program to print a message "Hello world" if the input character is 'Y'

(3)

```
int main()
{
  char ch;
  printf("Enter the character");
  scanf("%c", &ch);
  if (ch == 'Y')
  {
    printf("Hello world");
  }
  return 0;
}
```

(2) If else statement :- It is used to execute one block of code if the condition is true and another block of code if the condition is false

Syntax :-

```
if (Condition)
{
    Block of code.
}
else
{
    Block of code
}
```

Example :- (1) write a c program to find the largest of two number.

```
int main()
{
    int a, b;
    printf("Enter the value of a & b");
    scanf("%d %d", &a, &b);
```

```
if (a>b)
{
    printf("a is larger");
}
else
{
    printf("b is larger");
}
return 0;
}
```

2) Write a c program to find given number is odd or even.

```
int main()
{
 int num;
printf("Enter the number");
scanf("%d", &num);
if (num%2==0)
{
  printf("Even");
}
else
{
  printf("odd");
}
return 0;
}
```

③ else if ladder :- It is multiway branching statement- use to execute one block & code among multiple block & codes.

Syntax :-

```
if (condition ①)
{
    Block & code ①;
}
else if (condition ②)
{
    Block & code ②;
}
    .
    .
    .
    .
    .
    .
else
{
    Block & code (n);
}
```

Example :→ write a c program to find given number is positive, Negative or zero.

(7)

```
int main()
{
 int num;
 printf("Enter the number");
 scanf("%d", &num);
 if (num > 0)
  {
     printf("positive");
  }
 else if (num < 0)
  {
     printf("Negative");
  }
 else
  {
     printf("zero");
  }
 return 0;
}
```

④ Nested if :- A if inside another if is called nested if.

Syntax :-

```
if (condition①)
{
    if (condition②)
    {
        Block of code;
    }
}
```

Example ① C program to find given number is positive negative or zero using nested if statement.

```
int main()
{
 int num;
printf("enter the value of num");
scanf("%d", &num);
if (num >= 0)
{
    if (num > 0)
    {
```

```
        printf("positive");
    }
    else
    {
        printf("zero");
    }
}
else
{
    printf("Negative");
}
return 0;
}
```

(2) write a c program to accept 2 number & one operator as input & perform the corresponding arithmetic operation using else if ladder.

(3) write a c program to accept day number [1, 2, 3, 4, 5, 6, 7] & print the corresponding day using else-if ladder.

(5) Switch :- switch is multiway branching decision making statement & it only work with integer or characters.

* switch statement uses break to come out of the switch.

Syntax :-

integer (or) character

```
switch ( choice )
{
case choice 1: statement ;
               break ;
case choice 2: statement ;
               break ;
      .
      .
      .
default : statement
}
```

choice :- Here choice can be integer or constant.

if choice matches with any case choices [choice 1, choice 2 ....etc] the corresponding statement get executed & break will make the control to come out.

Switch is same as else-if ladder. Only difference is else-if ladder uses logical expression or relational expression, where switch use integer or character. to execute a particular block of code.

Example c program to perform basic Arithmetic operation using switch

```
int main()
{
  int n1, n2;
  char ch;
  printf("Enter the n1, operator & n2 in
          the same order");
  Scanf("%d.%c %d", &n1, &ch, &n2);

  Switch (ch);
  {
    Case '+' : printf("result=%d", n1+n2);
               break;
    Case '-' : printf("result=%d", n1-n2);
               break;
    Case '*' : printf("result=%d", n1*n2);
               break;
      :
      :
      :
```

```
        .
        .
        .
    default: printf("Enter valid operator");
}
return 0;
}
```

(2) Write a c program to accept number as input [1-7] and print corresponding day using switch

(3) write a c program to print grade based on the marks entered using else-if statement.

Marks $\geq$ 90 — 'A'

75 $\leq$ Marks $\leq$ 89 — 'B'

35 $\leq$ marks $\leq$ 74 — 'C'

else — 'fail'

* Justify for this program switch can not be applied (used)

Decision Making and looping

In c programming, many times problem involves performing the same task multiple times. writing the same code repeatedly (i.e multiple times) for such task is inefficient & makes program lengthy.

Instead of writing the same statements multiple times, write them once & execute multiple times using looping statements.

C provides three main type of looping statement-

- while loop.
- do-while loop.
- for loop.

① **while loop** :→ while is an entry-controlled loop statement.

Syntax :→

```
while (test-condition)
{
    Body of the loop
}
```

Test Condition is evaluated first and if the condition is true, then the body of the loop is executed, then again the test condition is evaluated if it is true, once again the body of the loop is executed. This process is continues until the test condition fails. if it fails the control comes out of the loop..

The program then continues with the statement immediatly below the loop.

Example ⇒

```
- - - -
sum=0;
n=1;
while (n<=10)
{
  sum = sum+n;  n=n+1;
}
printf ("sum=%d", sum);
- - - -
```

The above program performs sum of numbers from 1 to 10

Example write a c program to find the 'x' to the power of 'n' using while statement-

$$y = x^n$$

```
int main()
{
    int count, n;
    float x, y=1;
    printf("enter the value of x & n");
    scanf("%f %d", &x, &n);
    count = 1;
    while (count <= n)
    {
        y = y * x;
        count++;
    }
    printf("x to power n is = %f \n", y);
}
```

② do while loop:- The do while loop is called exit controlled loop.

Syntax

```
do
{
    Body of the loop
} while (test-condition);
```

Here the body of the loop is executed first, then the condition is checked if it is true, the program continues to execute the body of the loop. This process continues as long as condition is true, when condition becomes false the control comes out of the loop.

Note even condition fails first time, the body of the loop is executed once.

Example

write a c program to Read a number until user enter '0'

```
int main()
{
    int num, sum=0;
    do
    {
        printf("Enter the number (0 to stop): ");
        scanf("%d", &num);
        sum += num;
    } while(num!=0);
    printf("Total sum = %d \n", sum);
}
```

loop is executed until user enters '0'

Example

The body of the loop is executed until user enters '0' or negative number

```
do
{
    printf("Enter the input \n");
    scanf("%d", &num);
} while(num>0);
```

(3) For loop :⇒ for loop is also an entry controlled loop & is widely used when the number of repetitions is already known.

It combines, initialization, condition, and update in single line.

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

* Initialization of control variable is done in the begining. →(This is executed one time only)
  Example ⇒ i=0 or count=0.
* The value of the control variable is tested using "test-condition". the test-condition is relational Expression such as (or logical)
  i<=10 or count<=20.

If condition is true the body of the loop is executed otherwise the loop is terminated

(update) or decrement

* Increment, After the body of the loop is executed the control is transfered to for statement to increment (update) the control variable such i++ or count++.

Then new value of control variable is again tested using test-condition if condition satisfies again the body of the loop is executed.

* This process continues till the value of the variable fails to satisfy the test-condition.

Example

```
for(i=0; i<=9; i++)
{
    printf("%d\n", i);
}
```

flow of execution ① initialization → condition checked
② if condition true → body executes
③ After execution → update runs.
④ Again check condition → repeate (2,3,4) untill condition fails.

Example

Write a program to print only even numbers between range [2 to 10] using for loop

```
int main()
{
 int i;
 for(i=2; i<=10; i+=2)
   {
      printf("%d\n", i);
   }
 return 0;
}
```

Jumping statements :-

Normally, once the loop in C starts, it runs in sequence until its condition becomes false.

Jump statements in C are used to control the flow of a loop or to interrupt its normal execution. These statements allow program control to jump forward from one point another with in the code

C provides the following jumping statements

① **break** :- immediately terminates the loop or switch.

Example

```
for(i=0; i<=5; i++)
{
    if(i==3)
    {
        break;
    }
    printf("%d\n", i);
}
```

(outside loop)

The for loop here is written to print the number from 0 to 5. But the break statement is executed if i==3 and it brings the program control out of the loop.

∴ The above program prints numbers from 0 to 2.

o/p

```
0
1
2
```

(2) Continue :- Continue will make the control to execute with the next iteration of the loop.

inside loop the statement below the continue are not executed for that particular iteration.

Example

```
for (i=0 ; i<=5 ; i++)
{
    if (i==3)
    {
        continue;
    }
    printf("%d\n", i);
}
```

The for loop here is to print number from 0 to 5. But the continue statement here is executed if i==3 and this brings the control back to the next iteration...

∴ The printf statement is not executed for i==3 ∴ The output of the above example is 0 1 2 4 5

③ goto label :- Control jumps directly to the labeled statements.

```
        ———
        ———
    goto next;
        ———
        ———
        ———
        ———
next:   ———
        ———
        ———
```

(Control jumps to next) →→

} these statements are not executed

Example

```
int main()
{
    int i;
    for (i=0; i<=5; i++)
    {
        printf("%d\n", i);
        goto skip;
    }
skip: printf("loop ends after 1st iteration");
    return 0;
}
```

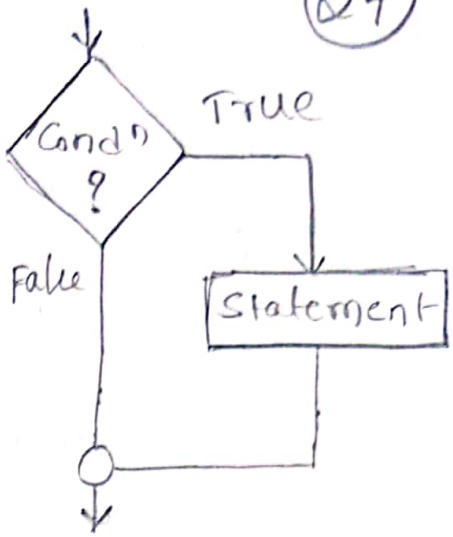
in this example the for loop is written to print numbers from 0 to 5. But goto is executed immediately after printf (prints first number ie 0) then the control jumps to the skip table and prints the corresponding message.

o/p

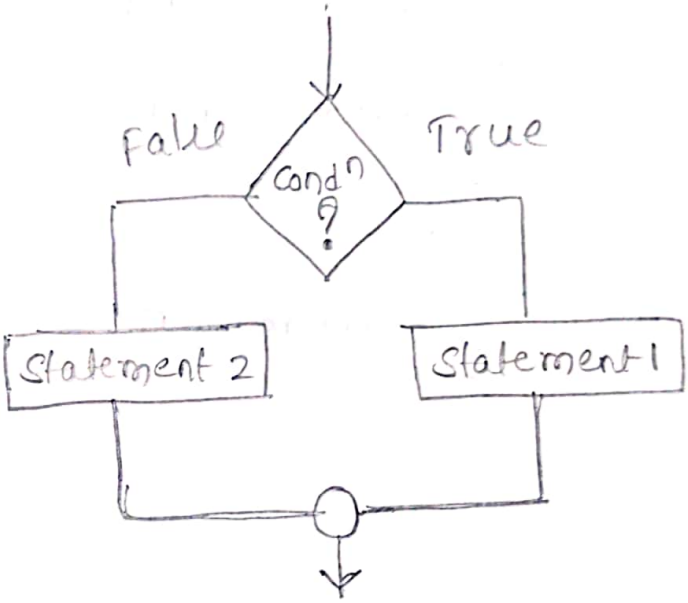
```
0
loop ends after 1st iteration.
```

Syntax and flowchart-

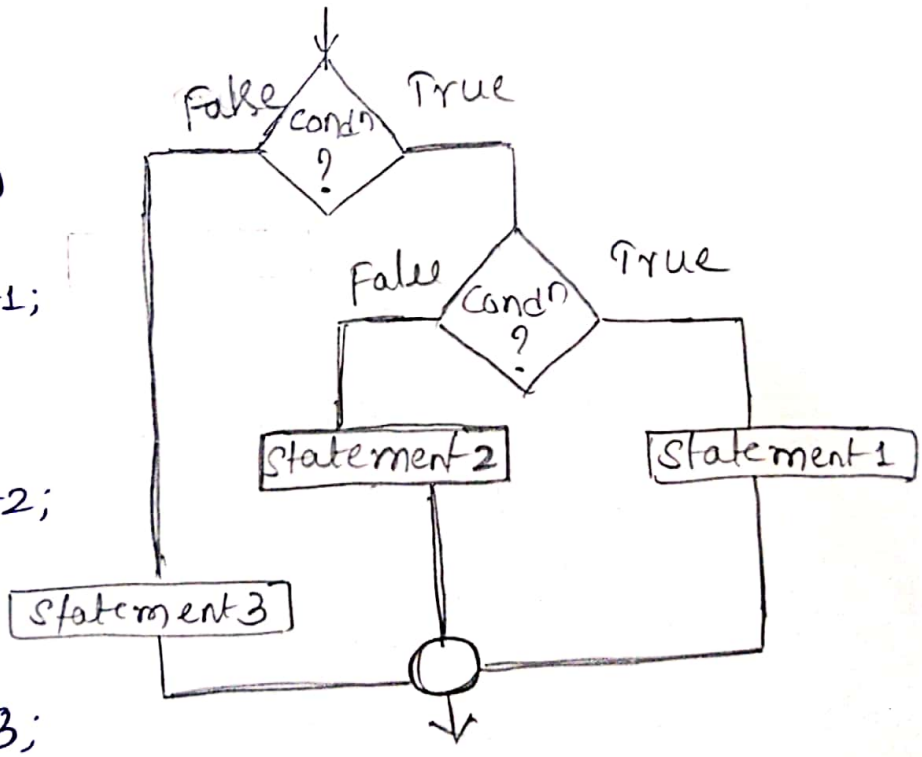
①
```
if (condition)
{
   statement-;
}
```



②
```
if ( condition )
{
   statement-1 ;
}
else
{
   statement-2;
}
```

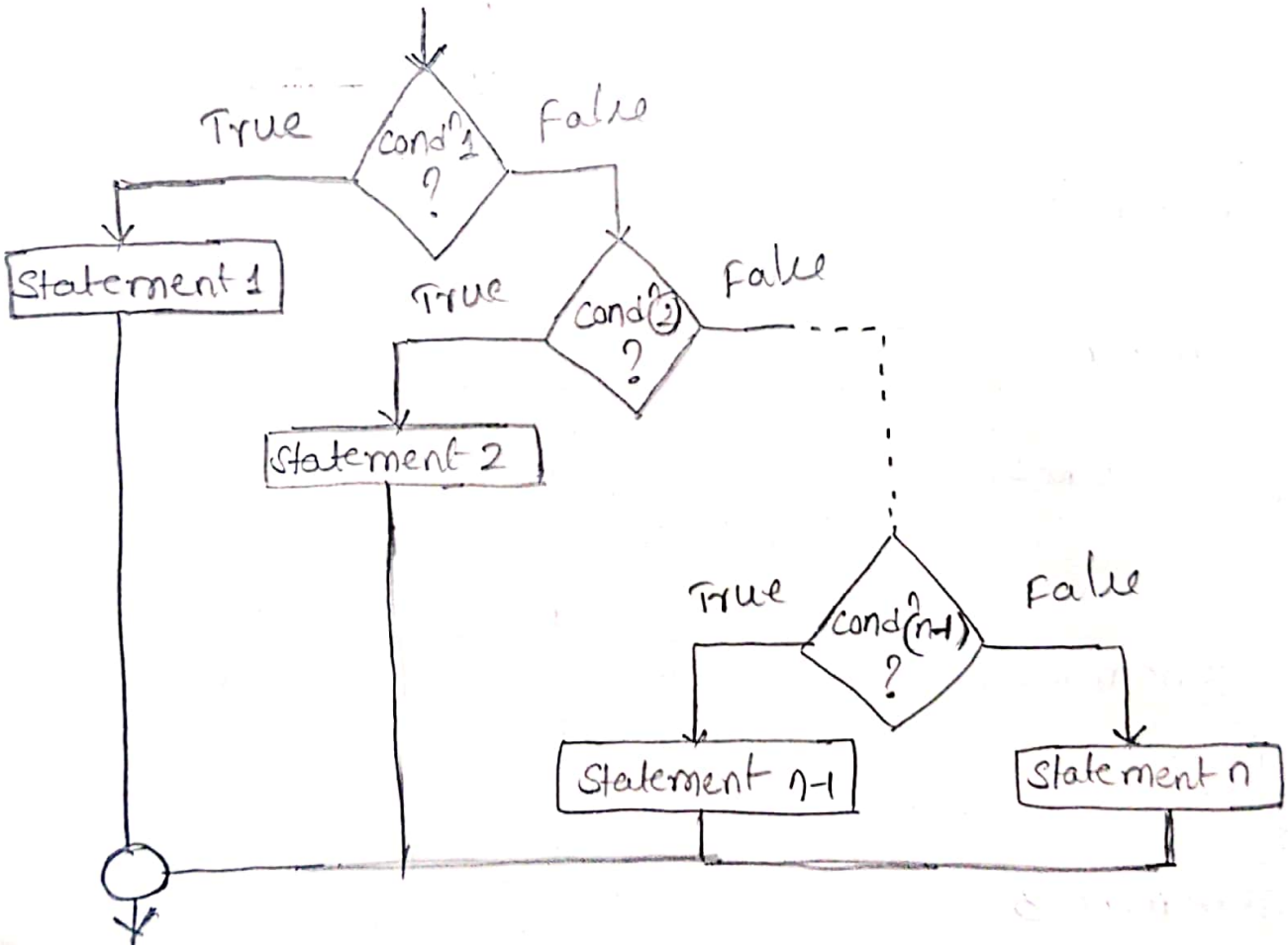


③
```
if (Condition 1)
{
   if(condition 2)
   {
      statement-1;
   }
   else
   {
      statement-2;
   }
}
else
{
   statement 3;
}
```



④

```
if (condition 1)
{
    statement 1;
}
else if (condition 2)
{
   statement 2;
}
      .
      .
      .
else if (condition n-1)
{
   statement n-1;
}
else
{
    statement n;
}
```

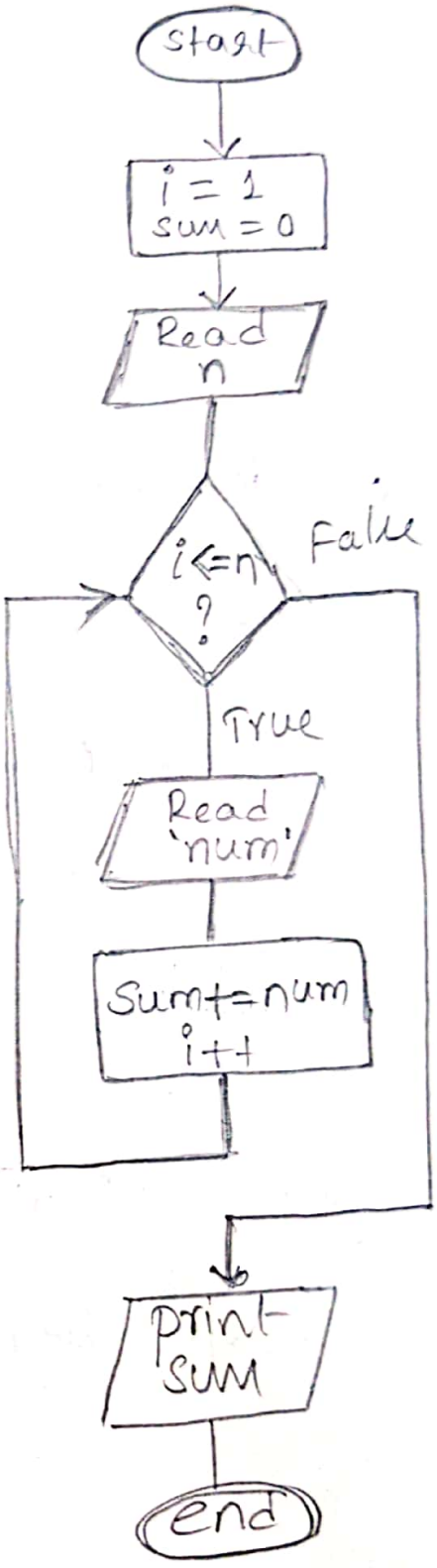


Example program with flow chart-

(1) Write a c program to add "n" number

(a) using while loop :-

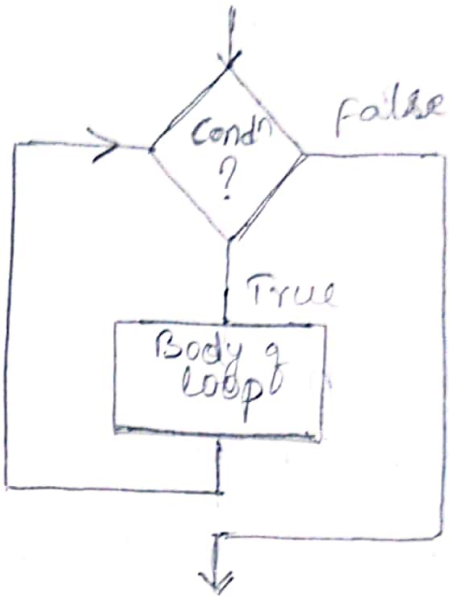
```
main( )
{
  int n, i=1;
  float num, sum=0;
  printf("Enter the value of 'n');
  Scanf("%d", &n);
  while (i<=n)
  {
    printf("Enter the number:");
    Scanf("%f", &num);
    sum+=num;
     i++
  }
printf("sum =%f", sum);
}
```



Looping statements :-

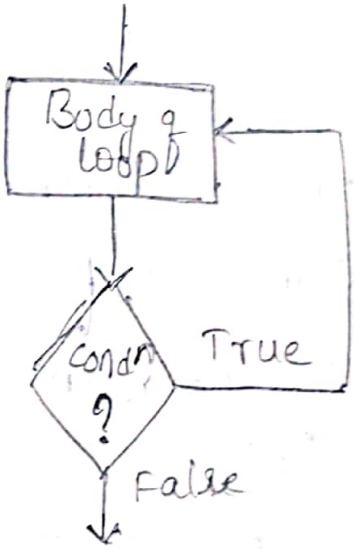
① while loop

```
while (cond^n)
{
    Body of loop;
}
```



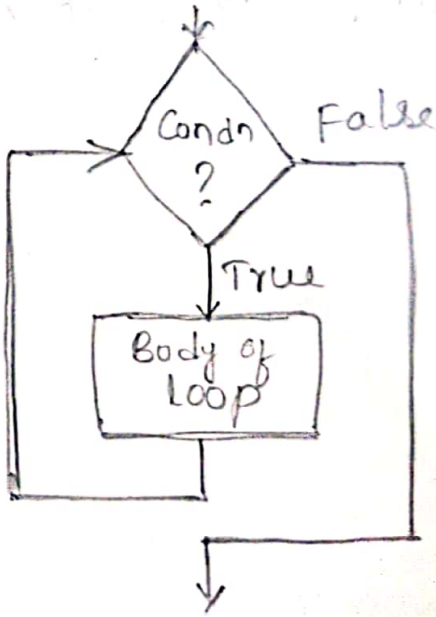
② do while

```
do
{
    Body of loop;
} while (cond^n);
```



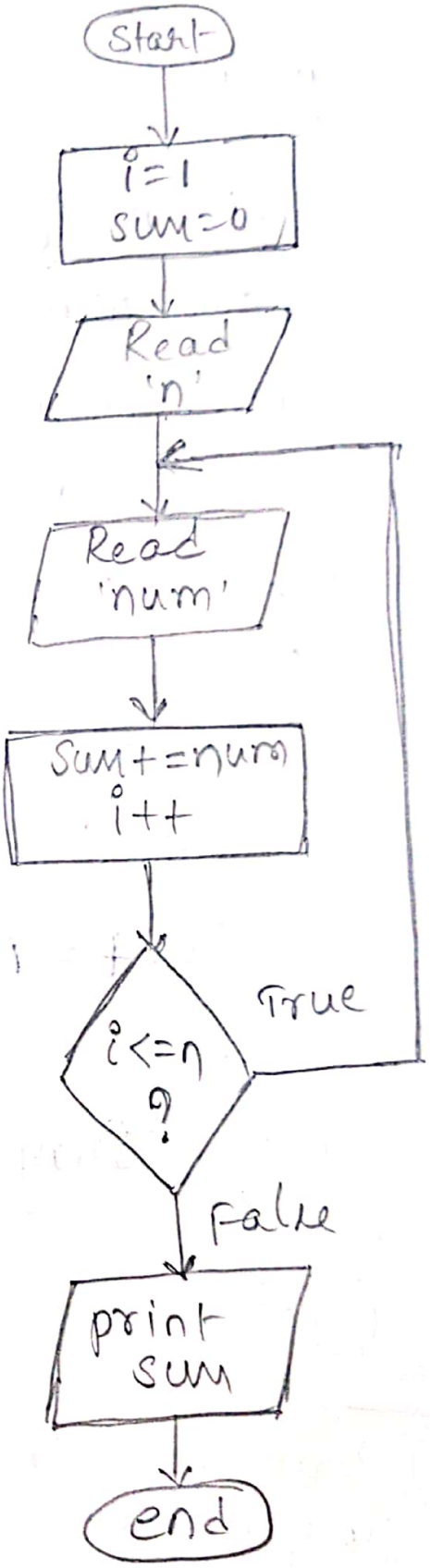
③ for loop :-

```
for(initialization ; condition ; update)
{
    Body of loop;
}
```



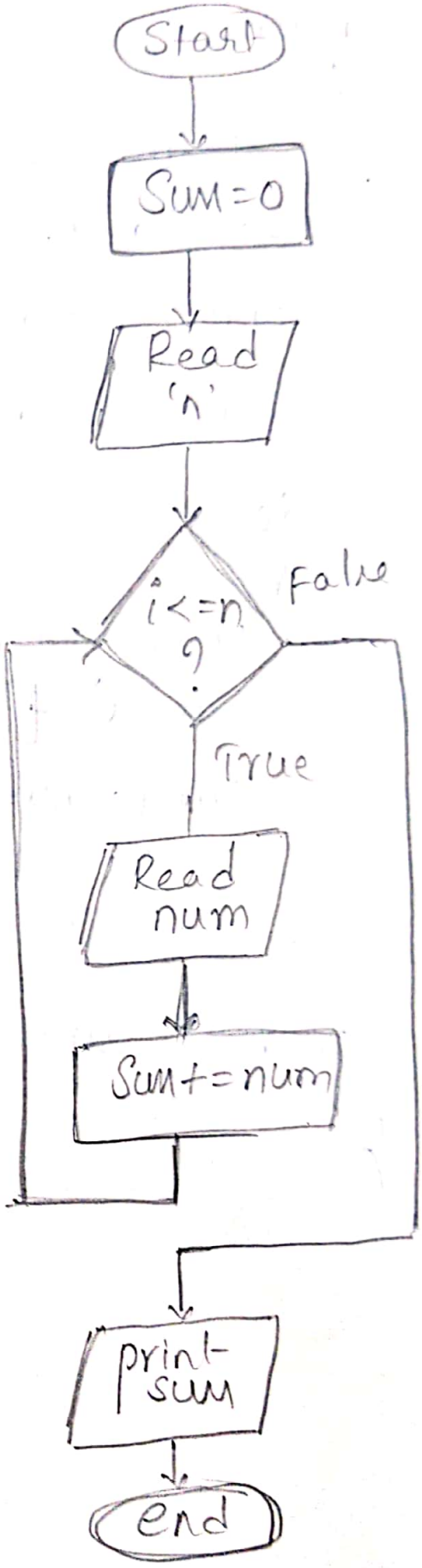
(b) using do while :-

```
main()
{
    int n, i=1;
    float num, sum=0
    printf("Enter 'n');
    scanf ("%d", &n);

   do
   {
     printf("Enter number:");
     scanf ("%f", &num);
     Sum+= num;
       i++;
   } while( i<=n)

printf(" sum = %f", sum");

}
```



(c) using for loop :-

```
main()
{
  int n, i;
  float num, sum=0;
 printf("enter 'n' ");
 Scanf("%d", &n);
for(i=1; i<=n; i++)
 {
    printf("enter number:");
    Scanf("%f", &num);
    Sum += Num;
 }
printf("Sum = %f", sum);
}
```



(2) Write a C program to find average of 'n' number
(*) using while, do while & for loop.
(*) Draw flow chart for each.