

Practical-5

PROF. SHRADDHA PATEL

Practical:5.1

- Write a program to calculate sum of all digits of a number like this (Hint: if($\text{no}/10 == 0$) inside while loop):
- Enter any number: 121
- $1+2+1=4$
- Input=no(121) ---Process-logic-----output:sum($1+2+1$)=4

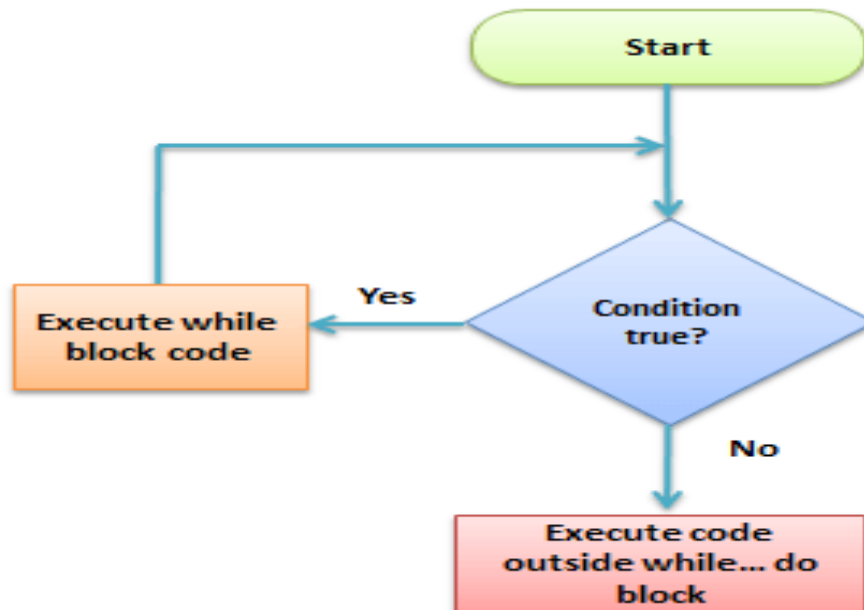
loop

- A **Loop** executes the sequence of statements many times until the stated condition becomes false.
- The purpose of the loop is to repeat the same code a number of times.

Types of loop

- In an **entry-controlled loop**, a condition is checked before executing the body of a loop.
- It is also called as a pre-checking loop.
- In an **exit-controlled loop**, a condition is checked after executing the body of a loop.
- It is also called as a post-checking loop.

Loop



- The control conditions must be well defined and specified
- otherwise, the loop will execute an infinite number of times.
- The loop that does not stop executing and processes the statements number of times is called as an **infinite loop**

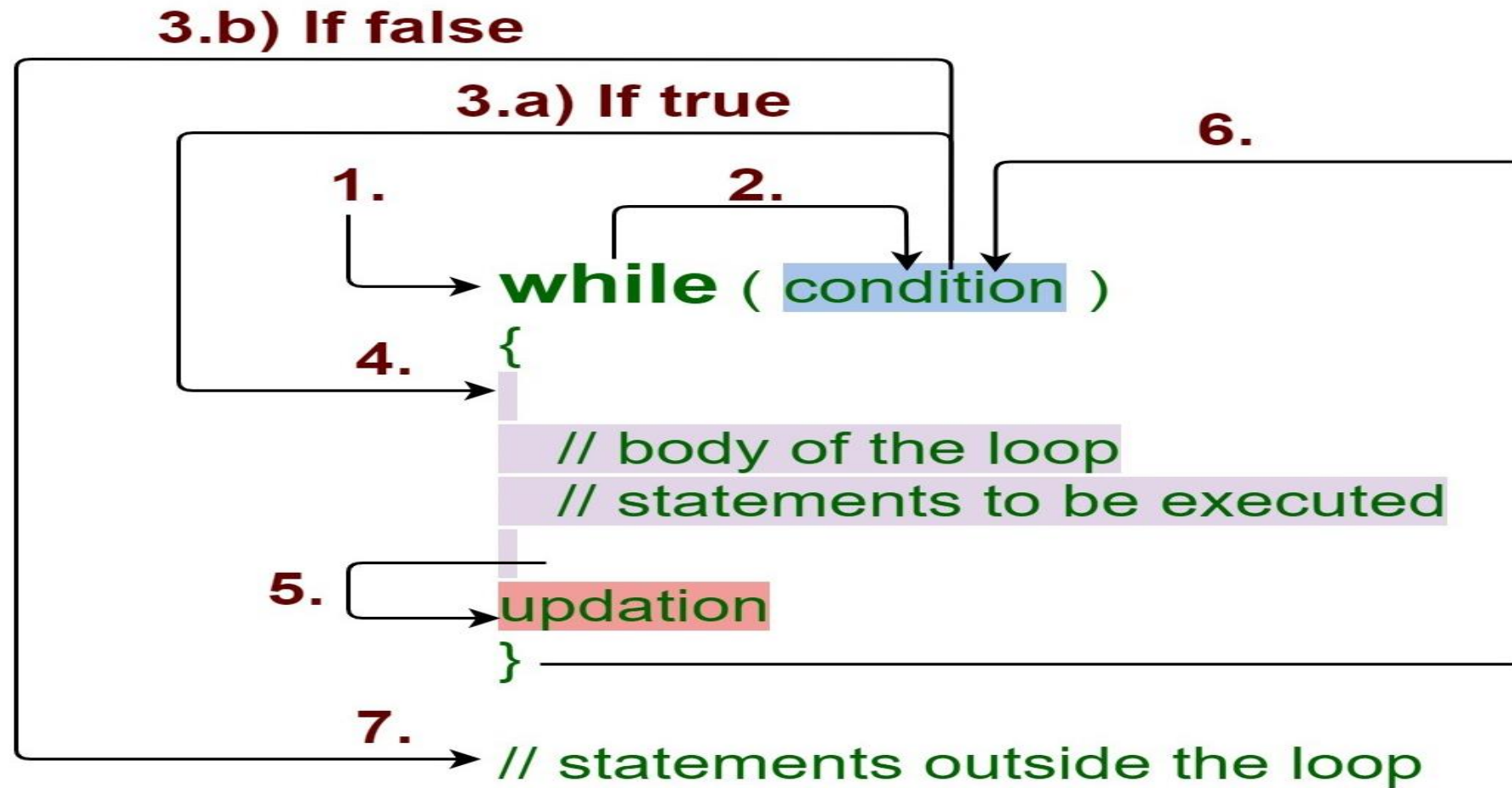
Types of Loop in C

- C' programming language provides us with three types of loop constructs:
 1. The while loop
 2. The do-while loop//exit-controlled
 3. The for loop

While Loop

- It is an entry-controlled loop.
- In while loop, a condition is evaluated before processing a body of the loop.
- If a condition is true, then and only then the body of a loop is executed.

While Loop



While Loop

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

```
int main(void)
```

```
{
```

```
    int j;
```

```
    j = -5;
```

```
    // while loop
```

```
    while(j <= 0)
```

```
    {
```

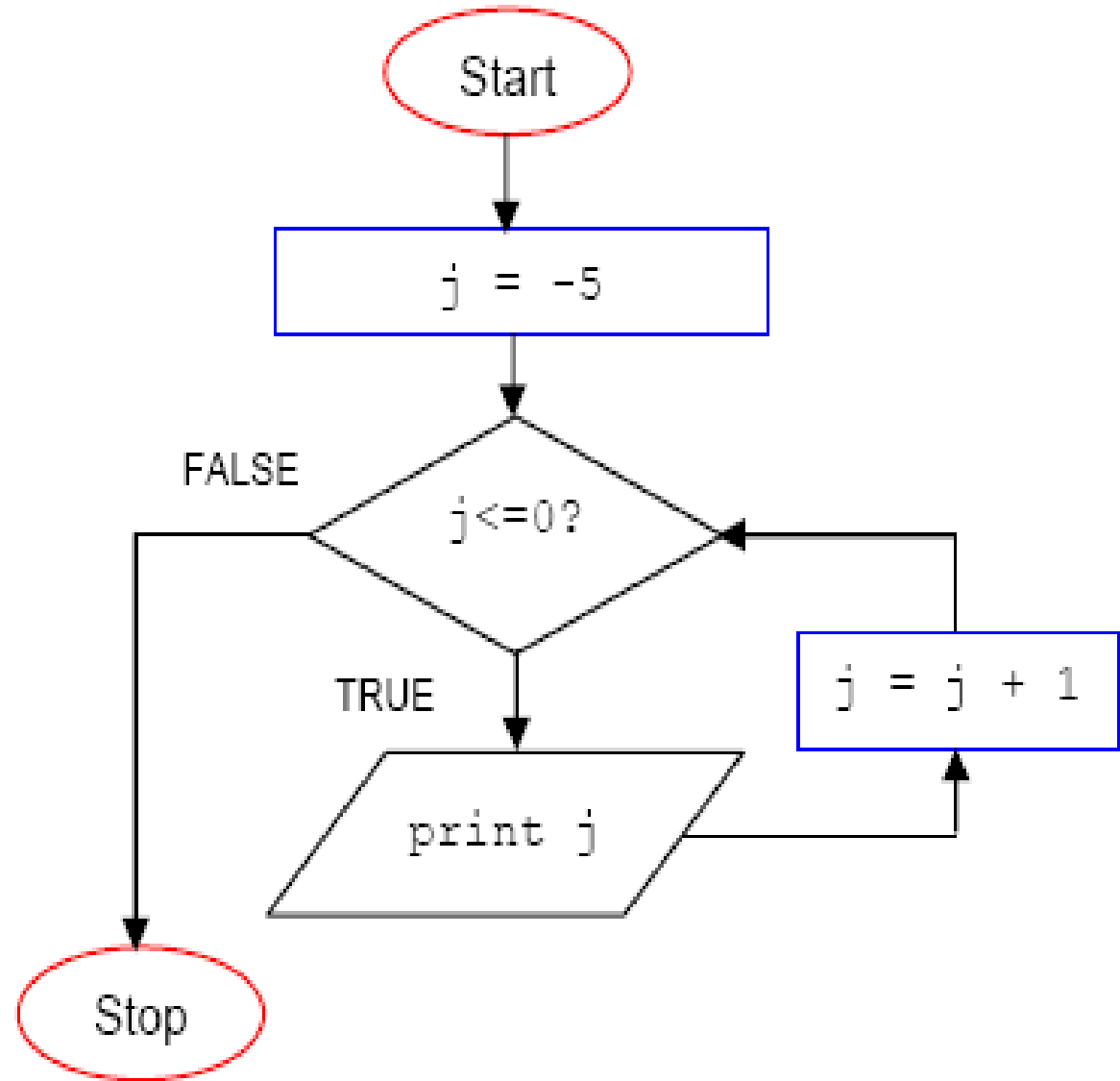
```
        printf("%d ", j);
```

```
        j = j + 1;
```

```
    }
```

```
    return 0;
```

```
}
```



Example

```
Void main()  
{  
    int i = 1; // initialization expression  
    while (i < 5) // test expression  
    {  
        printf("Hello World\n");  
        i++; // update expression  
    }  
}
```

Practical-5.1

1. Int num,sum=0
2. Enter no
3. Read number //345
4. While (no > 0) //
 1. Sum=sum+no%10;
 2. No=no/10
5. Print sum

345 > 0	34 > 0	3 > 0	0 > 0
true	true	True	False
Sum=0+345%10=5	Sum=5+34%10= 9	Sum=9+3%10=12	
No=345/10=34	No=34/10=3	No=3/10=0	

Practical-5.2

- Write the program to display the number in reverse order
- (e.g. 123-->321)
 - (a) Using printf () statement inside the loop.
 - (b) Using printf () statement outside the loop.

Practical-5.2

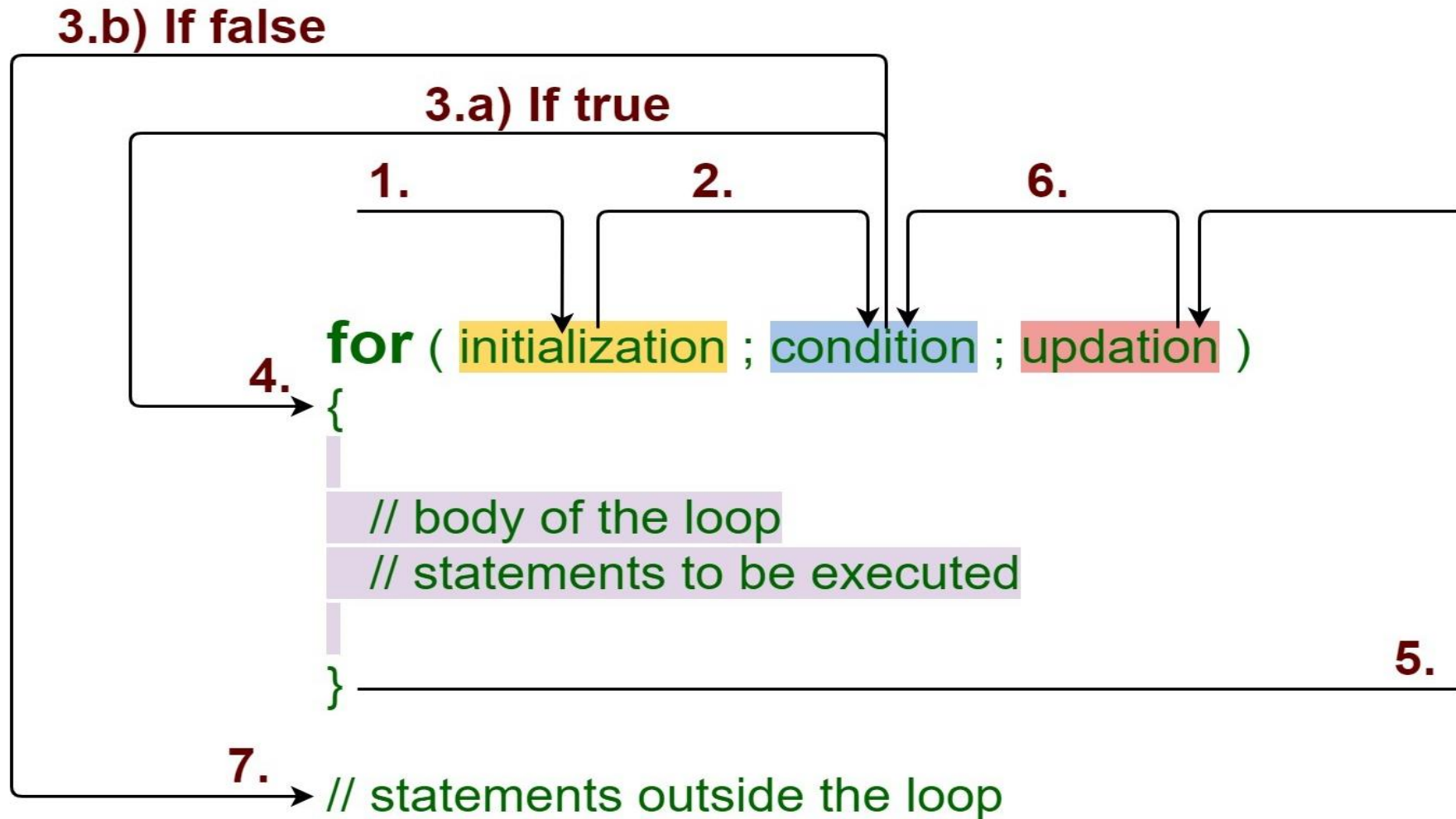
Printf inside loop

- While No > 0
 - {
 - Rev= no%10
 - Print rev
 - No=no/10
 - }

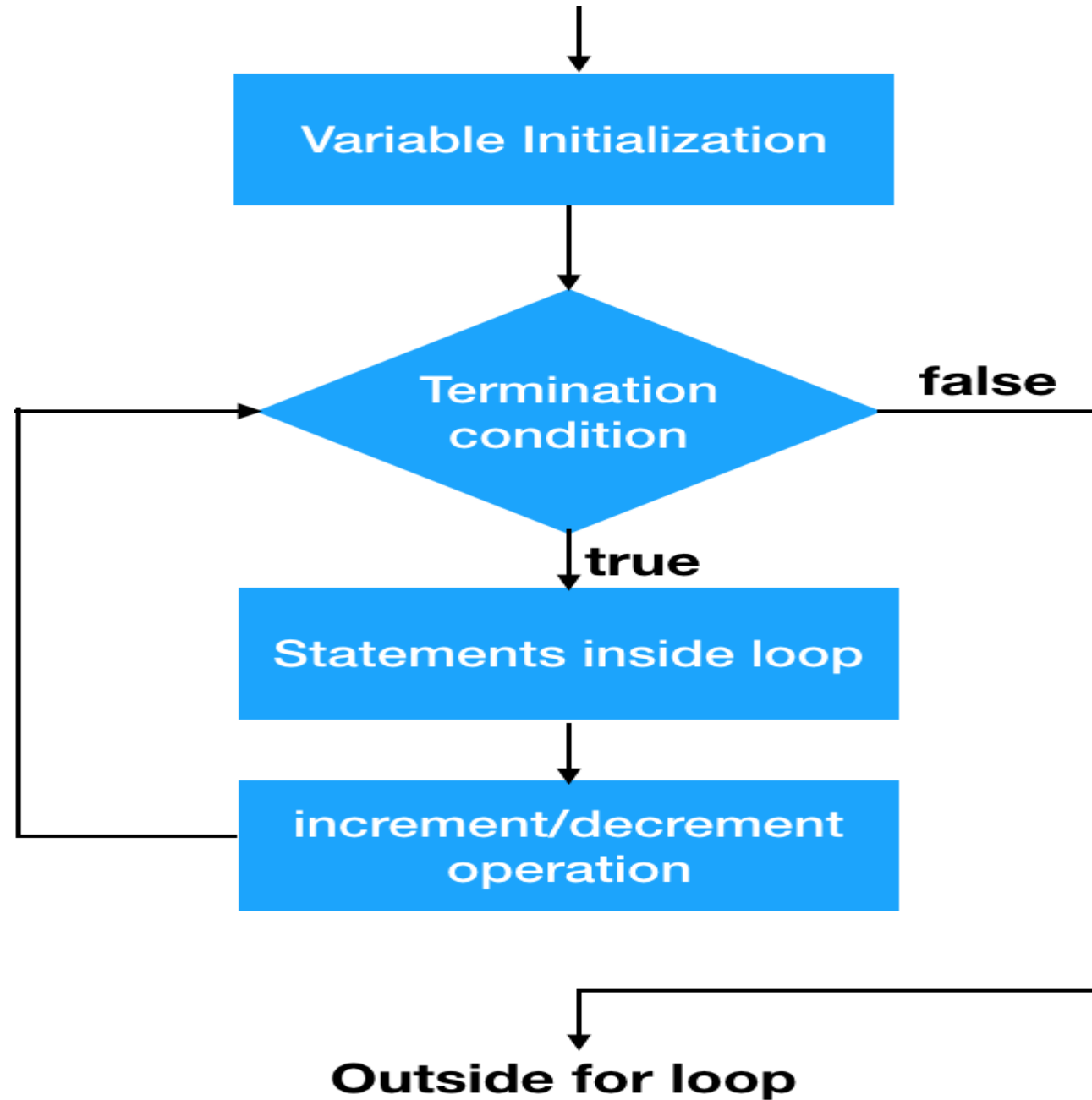
Printf outside of loop

- While no > 0
 - {
 - Rev=rev*10
 - rev=rev+no%10
 - No=no/10
 - }
- Print Reverse Number

For loop



For loop



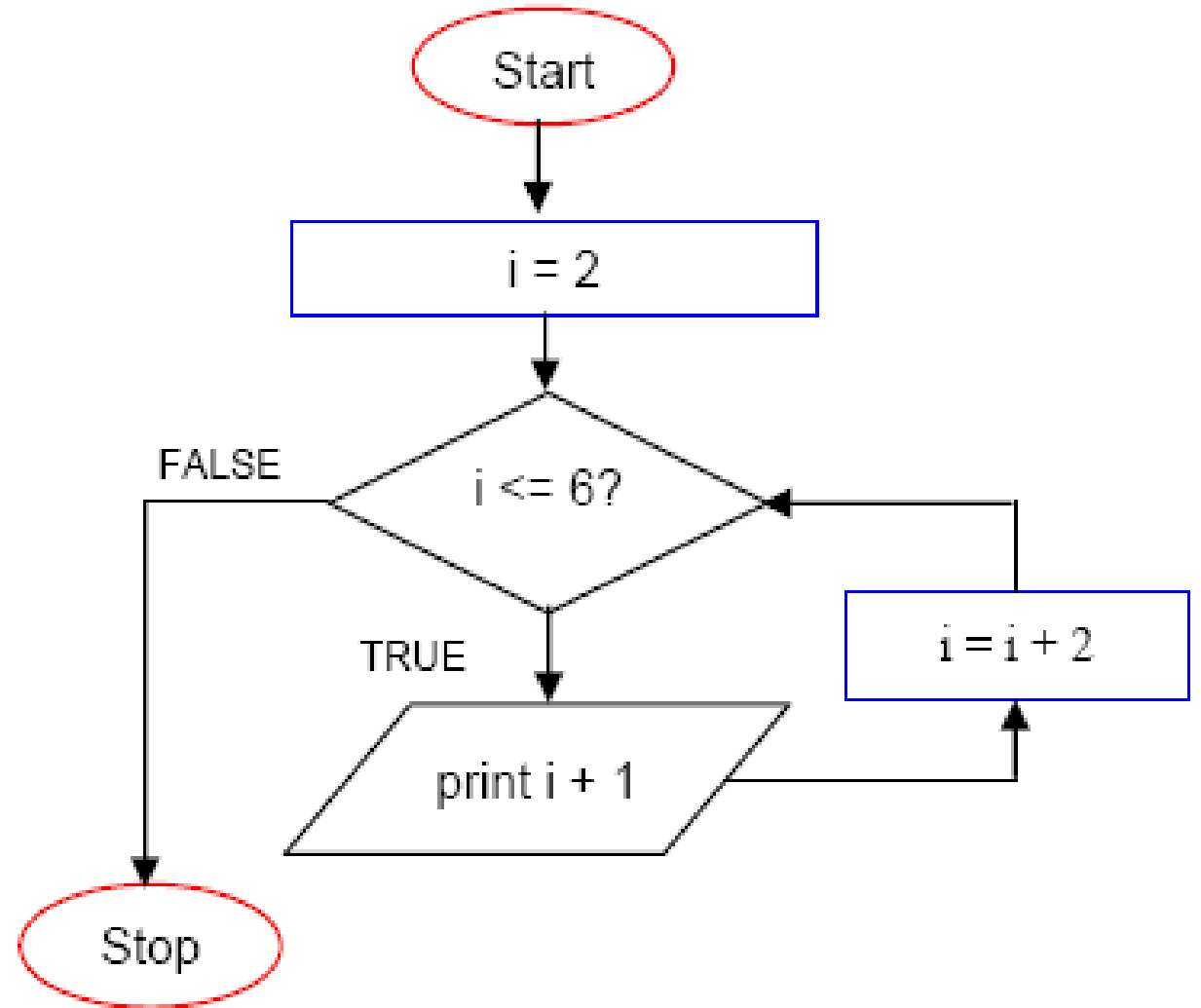
Example

```
for (i = 2; i <= 6; i = i + 2)
```

```
{
```

```
    printf("%d\t", i + 1)
```

```
}
```



Practical-5.3

Write a program to print following Series: $1 + 3 + 5 + \dots + n$

1. Initialization
2. Condition
3. Increment/decrement
4. Print i

Practical-5.4

Write a program to check that whether the entered number is prime or not.

A prime number (અભાજ્ય સંખ્યા/અવિભાજ્ય સંખ્યા) is a whole number greater than 1 that can only be divided by itself and 1.

E.g. 5 is prime number as 5×1 and 1×5

Practical-5.4

1. Int l,flag=0,no

2. i=2 , i<no ,i++

no%i==0

flag=1

break;

3. If(flag==1)

not prime

4. Else

prime

Practical-5.5

Write a program to print first N prime numbers

Practical-5.5

```
int main()
{
    int i=1, j=1, n=0, ct=0, Number;
    printf("Enter the Number\n");
    scanf("%d", &Number);
    while (n<Number)
    {
        ct=0;
        for(j=1; j<=i; j++)
        {
            if(i%j==0)
                ct++;
        }
        if(ct==2) //factor=2
        {
            printf("%d\t", i);
            n++;
        }
        i++;
    }
}
```

Practical-5.6

- Write a program to print Armstrong number (or Narcissistic no) between 0 to 999.
- Armstrong no: entered no. = $\{(1\text{st digit})^3 + (2\text{nd digit})^3 + (3\text{rd digit})^3 + \dots\}$
- e.g.
- 153 is Armstrong due to $1^3 + 5^3 + 3^3 = 153$
- use: `pow(a, x)`

Practical-5.6

- **Armstrong number** is a number that is equal to the sum of cubes of its digits.
- A number is an **Armstrong Number** or narcissistic number if it is equal to the sum of its own digits raised to the power of the number of digits.
- E.g 1634,153,371

Practical-5.6 [check Armstrong or not]

Armstrong no for n digit

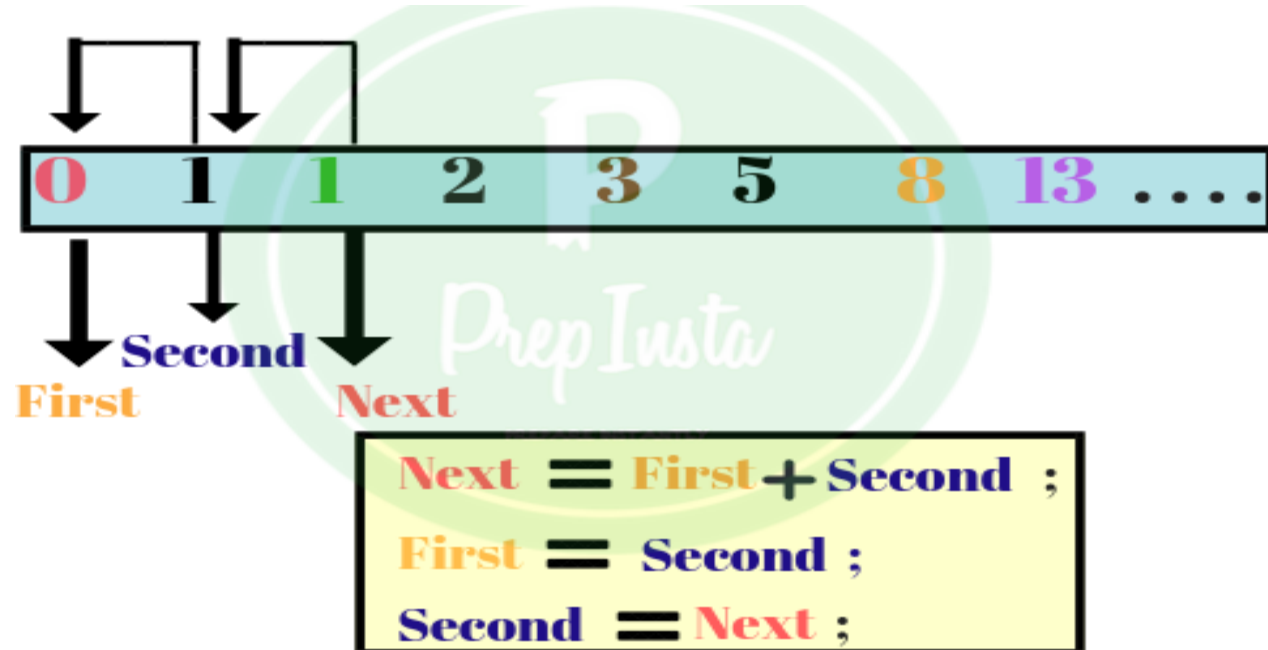
1. count the number of digits of the Number entered by user
 1. no=Eno
 2. While (no !=0) { No/=10 ; digit++;}
2. No=Eno
3. While(no!=0)
 1. rem=no%10 //Get individual digit 153
 2. Sum+=pow(rem,digit)
 3. No=no/=10
4. If(sum==no)
 1. Print Armstrong
5. Else
 1. Not Armstrong

Practical-5.6: Armstrong between 0 -999

1. Start//143
2. For loop
 1. Get first digit
 2. Get second digit
 3. Get third digit
 4. Calculate sum =pow(f1,3)+pow(s1,3)+pow(t1,3)
 5. Check sum== no
 6. Print no
3. End

Practical-5.7

- Write a Program to print Fibonacci series. (1 1 2 3 5 8 13.....)



Practical-5.7

1. start
2. Initialize variables
3. Read input value –n
4. Print first 2 no- first(n0) and second (n1)
5. While ($l < n$)
 1. Logic
 2. Print next
6. End

Practical-5.8

Write a program to print $1/1+1/2+1/3+1/4+\dots+1/N$ series.

1. Start
2. Initialize variable
3. Read no
4. While ($I \leq \text{no}$)/for loop
 - Print series
5. End

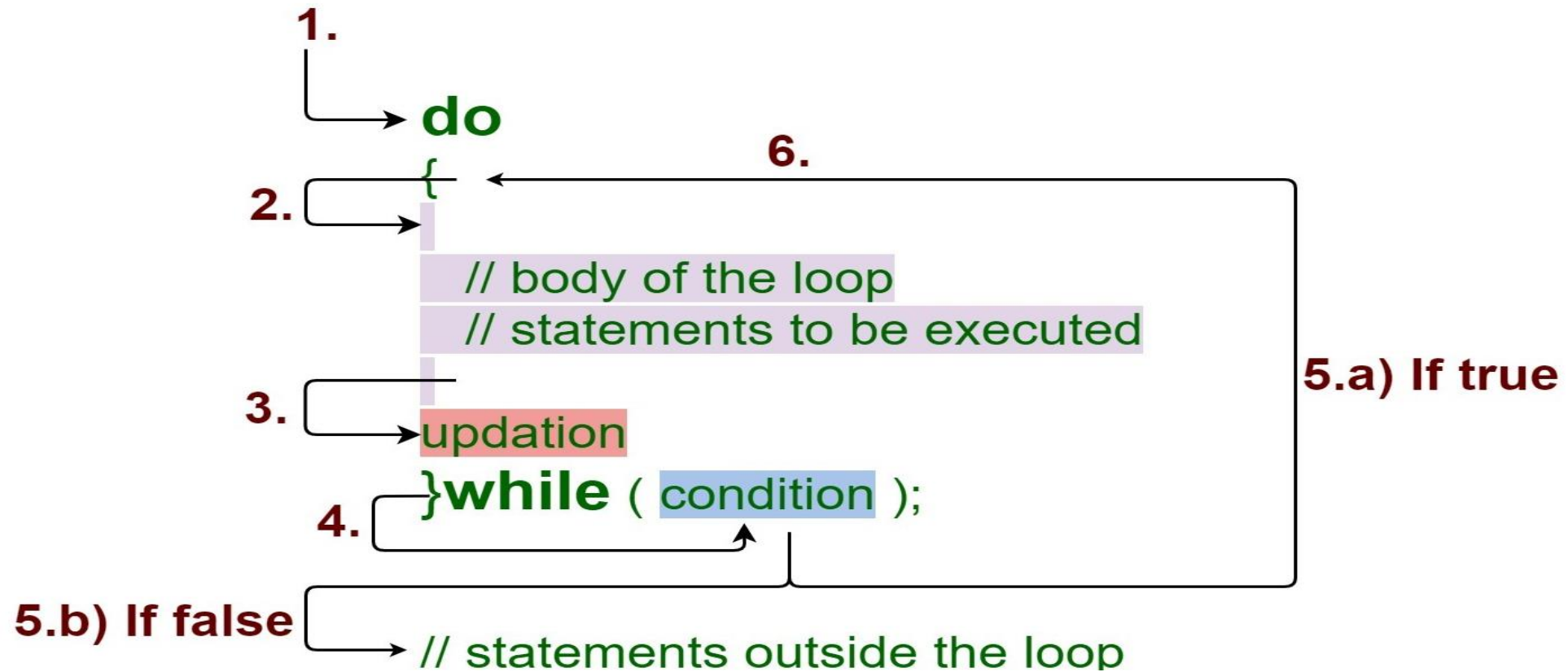
Practical-5.9

write a simple calculator program performing all five basic function shown below.

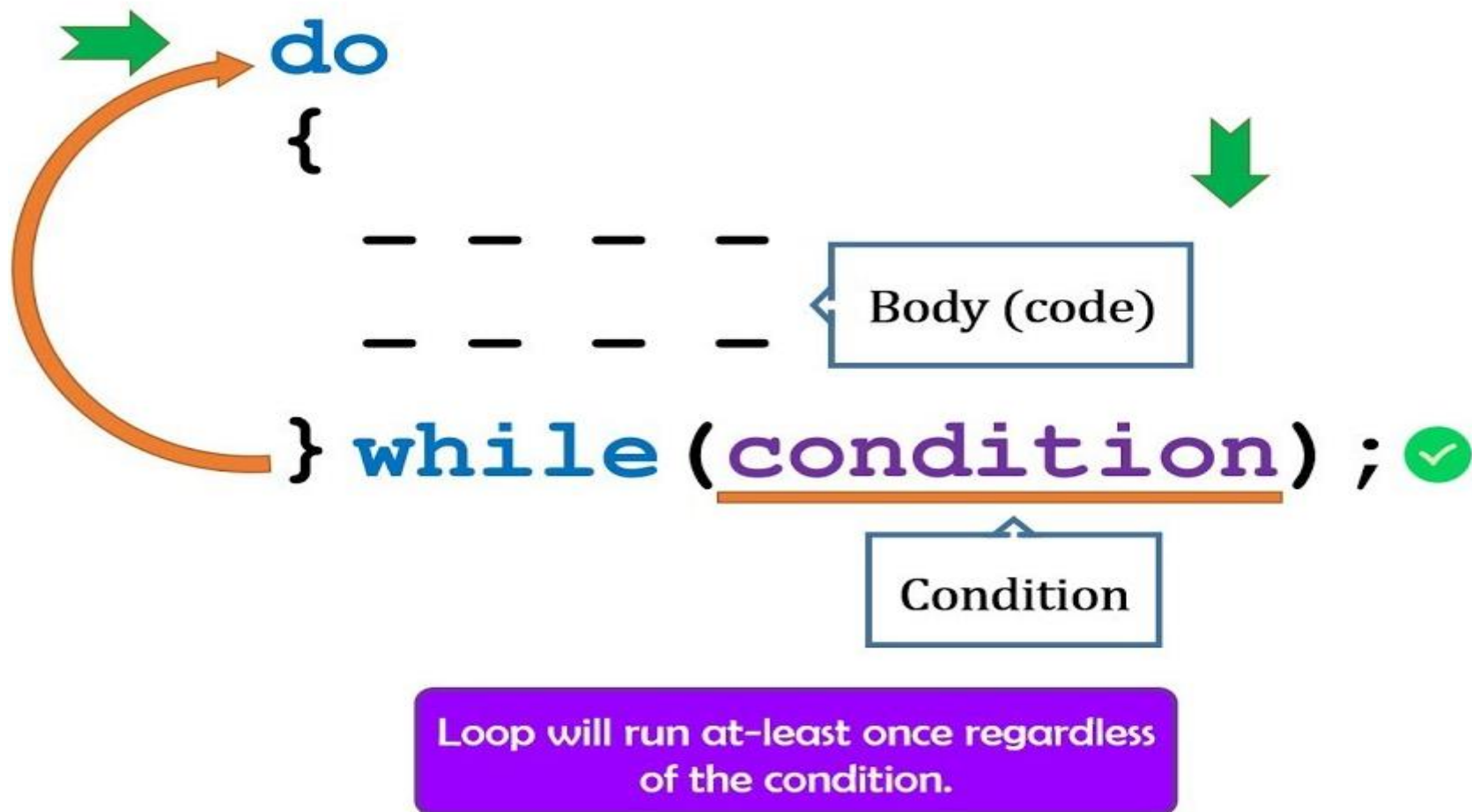
Make sure that after performing each operation your program should prompt to keep continue after each operation (do not use GOTO statement)

- Enter the value of a and b : 5 and 4
- Enter any operation listed below:
 1. Addition (press +)
 2. Subtraction (Press -)
 3. Multiplication (Press *)
 4. Division (Press /)
 5. Modulo (Press %)
- Enter your choice of operation: +
- The sum of 5 and 4 is : 9
- Do you want to continue (Y/N)? : Y

Do while loop

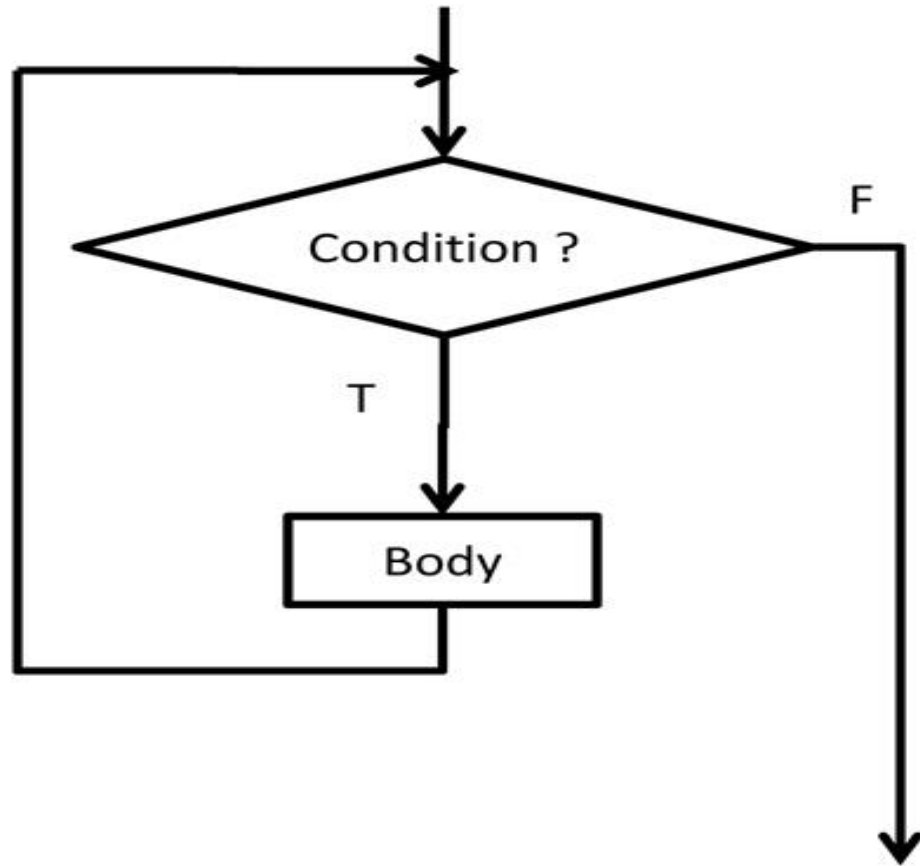


Do while loop (Exit-controlled loop)

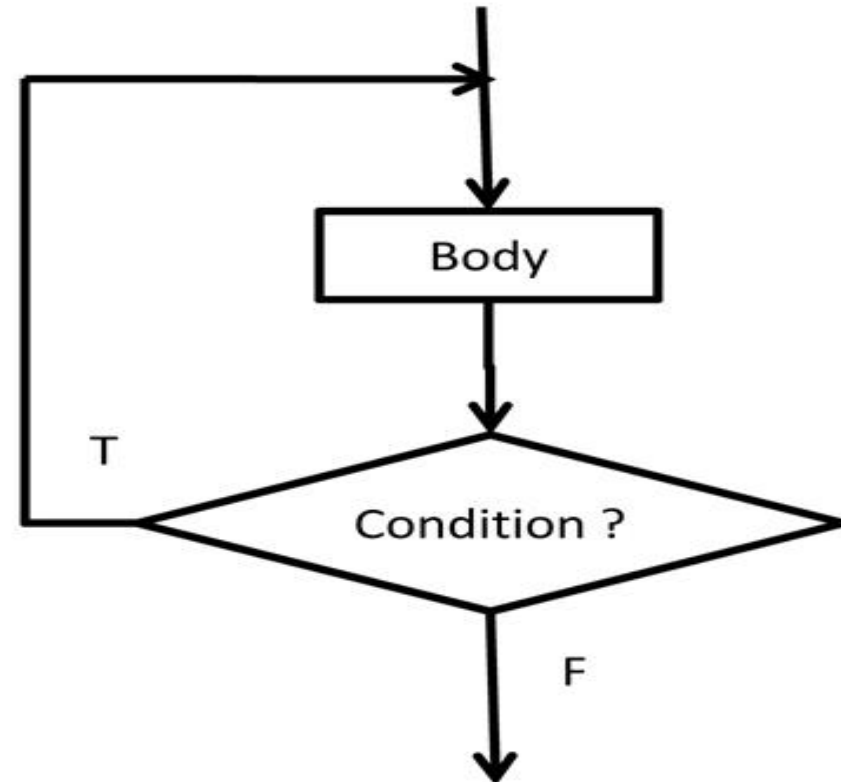


Do while loop

while(condition)
body;



do {
body;
} while(condition);



Do while loop

While

```
int i = 0;  
while(i > 0)  
{  
    printf("%d", i);  
    i--;  
}
```

do-while

```
int i = 0;  
do  
{  
    printf("%d", i);  
    i--;  
} while(i > 0);
```

Do while-Example

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

```
int main()
```

```
{
```

```
    int j = -5; // initialization
```

```
    do
```

```
    {
```

```
        printf("%d\n", j);
```

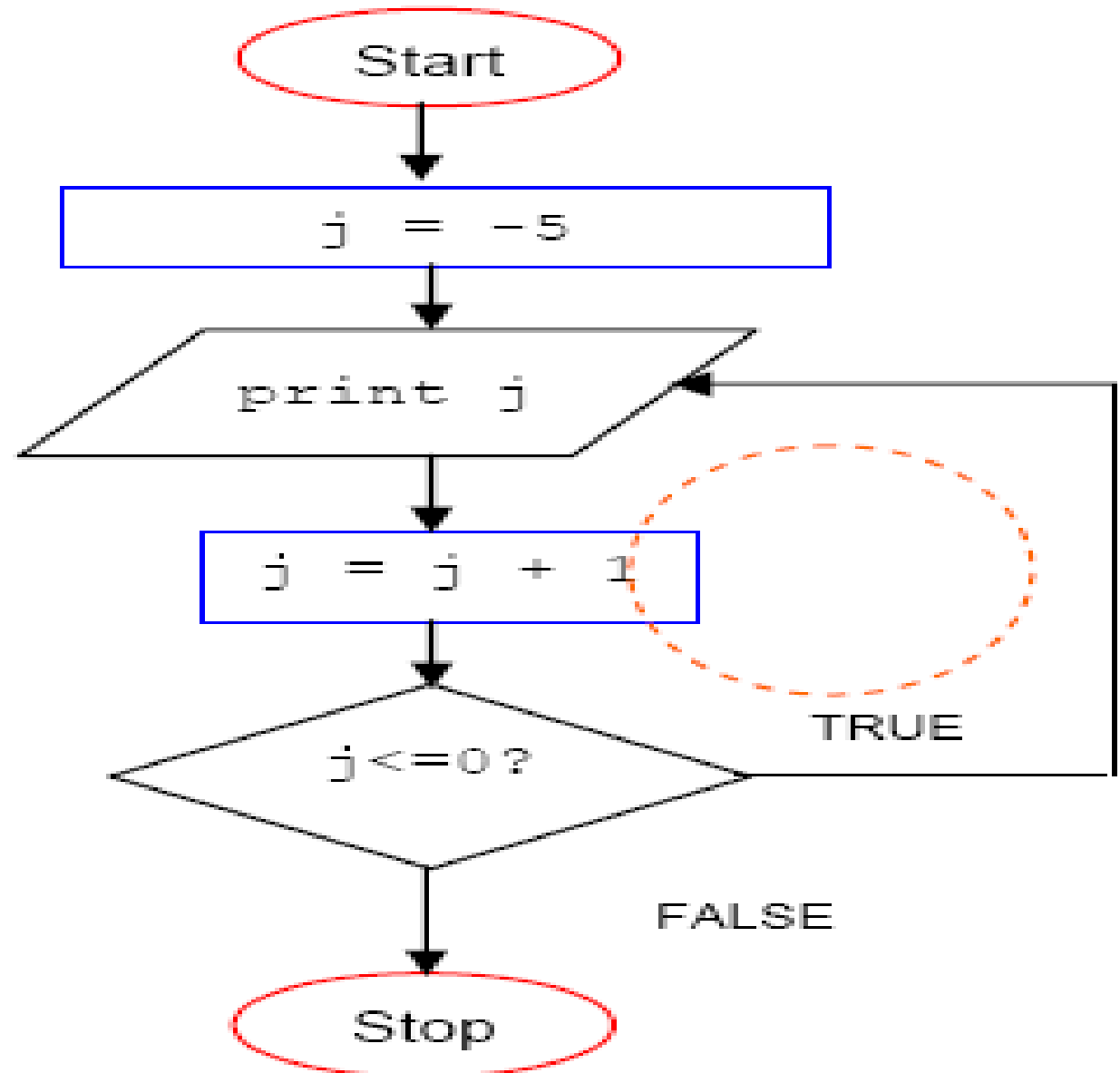
```
        j = j + 1;
```

```
    }
```

```
    while(j <= 0); // condition
```

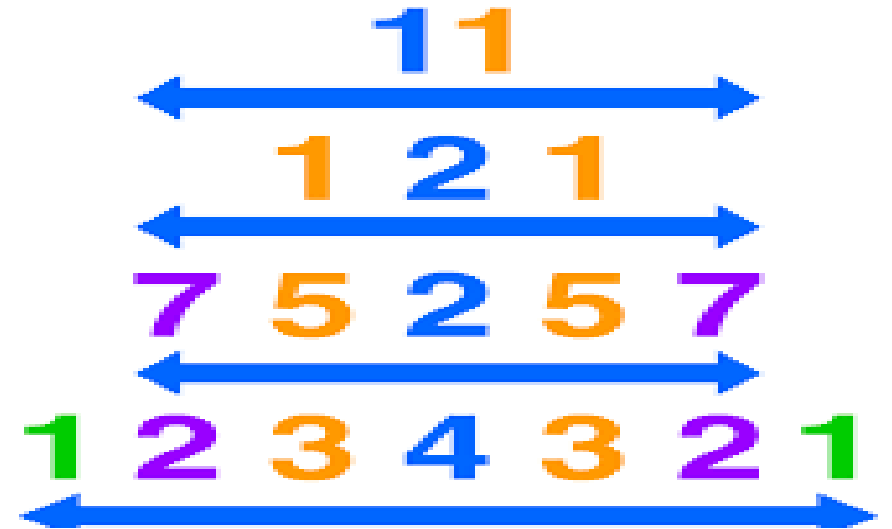
```
    return 0;
```

```
}
```



Practical-5.10

- Write a program to check that whether entered number is palindrome or not?
- Palindrome number: that remains the same when its digits are reversed.
- Original No= Reverse No



Practical-5.10

1. Start
2. Read no from user//65
3. Get Reverse of a Number(Exercise-5.2)//56
4. Check Entered No==Reverse No
5. Print Palindrome
6. Else
7. Not Palindrome

Practical-5.11

- Write a program to find out factorial (e.g. $4! = 4*3*2*1 = 24$) of any integer number using:

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

$$0! = 1$$

$$1! = 1$$

$$2! = 2*1$$

$$3! = 3*2*1$$

$$4! = 4*3*2*1$$

Practical-5.11

1. Start
2. Initialize variable(i=1 fact=1)
3. Read **no** from user //4
4. While (i<=**no**) //<=4
 1. Fact=fact*i //f=6*4
 2. i=i+1 //5
5. Print fact
6. End

Practical-5.12

- Write a program for each to print following patterns:

<pre>* * * * * * * * * *</pre>	<pre>1 2 2 3 3 3 4 4 4 4</pre>	<pre>1 0 1 1 0 1 0 1 0 1</pre>	<pre>4 3 2 1 3 2 1 2 1 1</pre>
<pre>1 2 3 4 5 6 7 8 9 10 11 - - - 15 . . 79 - - - - - 91</pre>	<pre>1 2 3 2 3 4 5 4 3 4 5 6 7 6 5 4 5 6 7 8 9 8 7 6 5</pre>	<pre>* * * * * * * * * * * * * * * * * * *</pre>	<pre>1 2 3 4 5 1 2 3 4 1 2 3 1 2 1</pre>

Practical-5.12

- If a loop exists inside the body of another loop, it's called a **nested loop**

```
while ( test) ← outer loop
{ //outer loop statements;
  while( test) ← inner loop
  {
    //inner loop statements;
  }
}
```

```
for ( initialization; test; update) ← outer loop
{ //outer loop statements;
  for( initialization; test; update) ← inner loop
  {
    //inner loop statements;
  }
}
```

```
do ← outer loop
{ //outer loop statements;
  do ← inner loop
  {
    //inner loop statements;
  } while (test);
} while(test);
```


Practical-5.12

```
for (int i = 0; i < rows; i++) ← Outer loop  
{  
    for (int j = 0; j < columns; j++) ← Inner loop
```

Example

Program to display the 5 asterisks in a row

```
Int j;  
for(int j=0; j<5; j++)  
{  
    printf("* ");  
}
```

OUTPUT:

1 2 3 4 5 asterisks



* * * * *

Example

- Program to display the 5 asterisks in a row (4 times)
- So need second "for" loop inside the first "for" loop:

```
Int i , j ;  
for( i=0;i<3;i++)  
{  
    for(j=0; j<5; j++)  
    {  
        printf("* " );  
    }  
}
```

OUTPUT:

* * * * *

Example

- Program to display the 5 asterisks in a row (4 times)
- So need second "for" loop inside the first "for" loop:

```
Int i , j ;  
for( i=0;i<3;i++)  
{  
    for(j=0; j<5; j++)  
        printf("* ");  
    printf("\n");  
}
```

OUTPUT:

The output consists of three rows of five asterisks each, separated by newlines. The first row is labeled "1st time", the second row is labeled "2nd time", and the third row is labeled "3rd time".

```
* * * * *  
* * * * *  
* * * * *
```

Example

```
Int main()
```

```
{
```

```
    int weeks = 3; int days = 4,i , j;
```

```
    for (i = 1; i <= weeks; i ++)
```

```
    {
```

```
        printf("Week: %d " , i );
```

```
        for (j = 1; j <= days; j++)
```

```
        {
```

```
            printf (" Day: %d" , j);
```

```
        }
```

```
        Printf("-----\n");
```

```
    }
```

```
}
```

```
Week: 1
```

```
Day: 1
```

```
Day: 2
```

```
Day: 3
```

```
Day: 4
```

```
-----
```

```
Week: 2
```

```
Day: 1
```

```
Day: 2
```

```
Day: 3
```

```
Day: 4
```

```
-----
```

```
Week: 3
```

```
Day: 1
```

```
Day: 2
```

```
Day: 3
```

```
Day: 4
```

```
-----
```

Practical-5.12.1

1. Total Row=4 - `for(i=0;i<5;i++)`

- Row-1(i) = Number of Star-1 (j=1)
- Row-2= Number of Star-2 (j=1,2)
- Row-3=Number of Star-3 (j=1,2,3)
- Row-4 =Number of Star-4 (j=1,2,3,4)



`for(j=0;j<=i;j++)`

1. `For(j=0 ; j < = i ; j++)`

1. Print *

2. End inner loop

2. `Print new line`

2. `End outer loop`

Practical-5.12.2

Pattern

Value of I=row	Value of j			
1				
2				
3				
4				

value in Patten

Value of I=row	Value of j=1	J=2	J=3	J=4
1	1			
2	2	2		
3	3	3	3	
4	4	4	4	4

Practical-5.12.3

Pattern

Value of I=row	Value of j			
1				
2				
3				
4				

value in Patten

Value of I=row	Value of j=1	J=2	J=3	J=4
1	1			
2	0	1		
3	1	0	1	
4	0	1	0	1

Practical-5.12.3

```
int i,j;
    for(i=0;i<5;i++)
    {
        for(j=0;j<=i;j++)
        {
            if(( i + j)%2==0)
                {   printf("1"); }
            else
                {   printf("0"); }
        }
        printf("\n");
    }
```

Practical-5.12.4

Pattern

Value of l=row	Value of j=1	J=2	J=3	J=4
1				
2				
3				
4				

Value in Patten

Value i= increase

Value J= decrease

Value of l=row	J=5	J=4	J=3	J=2	Value of j
1	4	3	2	1	4
2	3	2	1		3
3	2	1			2
4	1				1

Practical-5.12.5

Print the Following Floyd's triangles

Pattern

Value of $i = \text{row}$	Value of j			
1				
2				
3				
4				

value in Patten

Value of I=row	V...alue of j=1	J=2	J=3	J=4			
1	1						
2	2	3					
3	4	5	6				
4	7	8	9	10			
5	11	12	13	14	15		
6	16	17	18	19	20	21	
7							
.....							91

Practical-5.12.6

- Matrix of 5*9

I/J	1	2	3	4	5	6	7	8	9
1					*				
2				*	*	*			
3			*	*	*	*	*		
4		*	*	*	*	*	*	*	
5	*	*	*	*	*	*	*	*	*

Need 3 For Loop

1. Row
2. Space
3. Star/Number

Practical-5.12.6

- Matrix of 4*9

I/J	1	2	3	4	5	6	7	8	9
1					*				
2				*	*				
3			*	*	*				
4		*	*	*	*				
5	*	*	*	*	*				

Space

R=1 SP=5

R=2 SP=4

R=3 SP=3

R=4 SP=2

R=5 SP=1

Practical-5.12.6

- Matrix of 4*9

I/J	1	2	3	4	5	6	7	8	9
1					*				
2				*	*				
3			*	*	*				
4		*	*	*	*				
5	*	*	*	*	*				

patten (no of star- *)

R=1 star=1

R=2 star=2

R=3 Star=3

R=4 Star=4

R=5 Star=5

Example

```
for(i=1;i<5;i++)//5-row for loop
{
    for(sp=1;sp<n; sp++) //loop for space //for(k=5;k>l;k--)
    {
        printf(" ");
    }
    n--;
    for(j=1;j<=i;j++) //for pattern
    {
        printf("*");
    }
    printf("\n");
}
```

Practical-5.12.6

- Matrix of 4*9

I/J	1	2	3	4	5	6	7	8	9
1					*				
2				*	—	*			
3			*	—	*	—	*		
4		*	—	*	—	*	—	*	
5	*	—	*	—	*	—	*	—	*

Pattern

R=1 star=1

R=2 star=1,2

R=3 Star=1,2,3

R=4 Star=1,2,3,4

R=5 Star=1,2,3,4,5

Example

```
for(i=1;i<5;i++)//5-row for loop
{
    for(sp=1;sp<n; sp++) //loop for space
    {
        printf(" ");
    }
    n--;
    for(j=1;j<=i;j++) //for pattern
    {
        printf("* ");//extra space after *
    }
    printf("\n");
}
```

Practical-5.12.6

- Matrix of 4*9

I/J	1	2	3	4	5	6	7	8	9
1					*				
2				*	*	*			
3			*	*	*	*	*		
4		*	*	*	*	*	*	*	
5	*	*	*	*	*	*	*	*	*

pattern $j < (2*i) - 1$

R=1 star=1

R=2 star=3

R=3 Star=5

R=4 Star=7

R=5 Star=9

R=9 star= $2*n-1$

Practical-5.12.6(Actual)

- Matrix of 4*9

I/J	1	2	3	4	5	0	1	2	3
1					1				
2				2	3	2			
3			3	4	5	4	3		
4		4	5	6	7	6	5	4	
5	5	6	7	8	9	8	7	6	5

For loop

1. Rows

2. Triangle-1 -space

3. Triangle-2-pattern

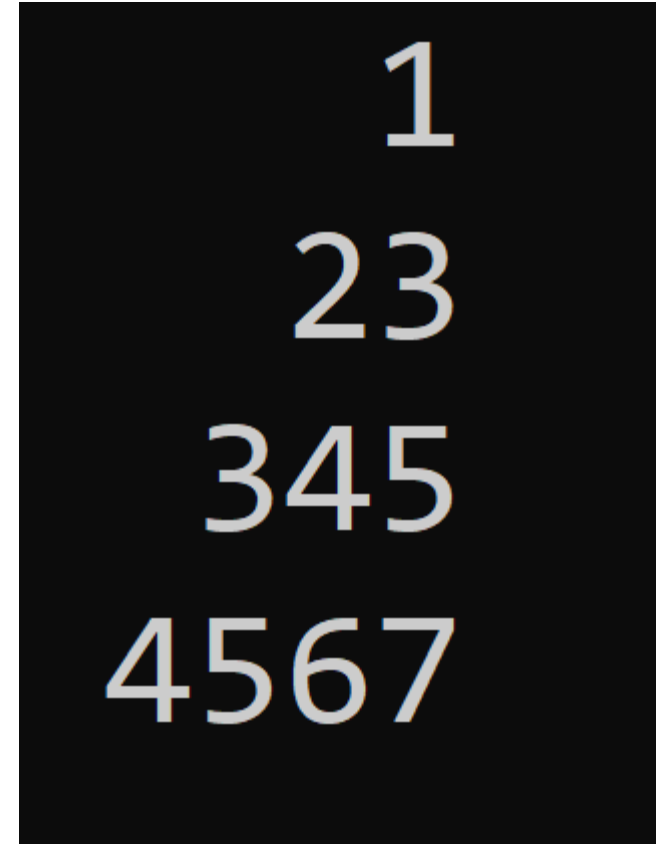
4. Triangle-3-pattern

Program

```
for(i=1;i<5;i++)  
{  
    TRIANGLE-1  
    for(sp=1;sp<n;sp++) //n=5  
    {  
        printf(" ");  
    }  
    n--;
```

Program

```
for(i=1;i<5;i++)  
{  
    TRIANGLE-1  
    TRIANGLE-2  
    for(j=0 ;j < i ; j ++)  
    {  
        printf("%d", i + j);  
    }  
}
```



1
23
345
4567

Program

```
for(i=1;i<5;i++)
```

```
{
```

```
    TRIANGLE-1
```

```
    TRIANGLE-2
```

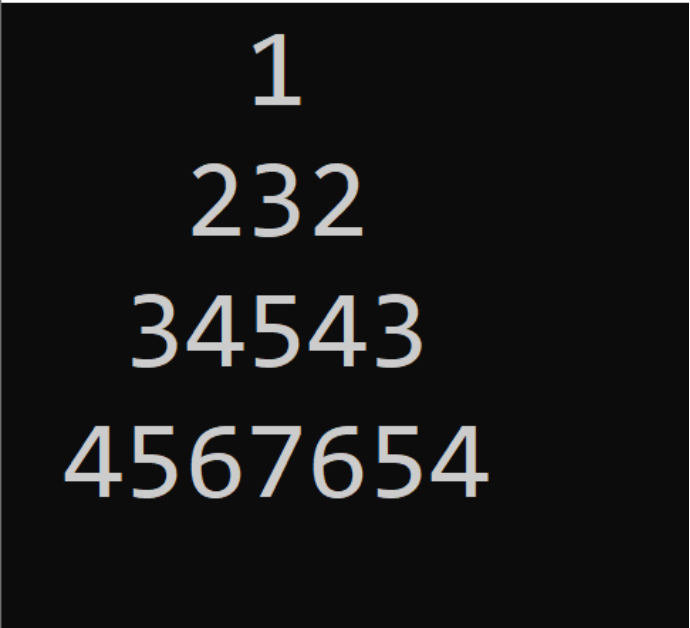
```
    TRIANGLE-3
```

```
        for(j=i-1 ;j > 0 ; j - - )
```

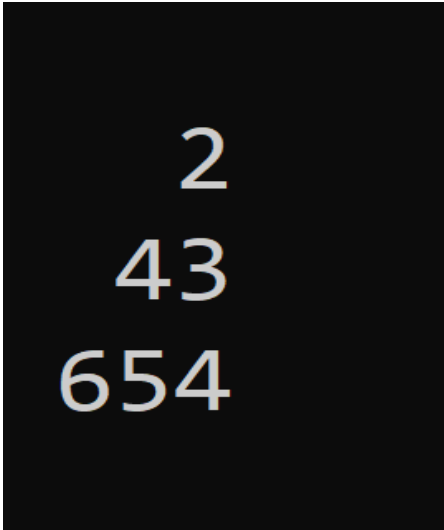
```
        {
```

```
            printf("%d", i + j -1);
```

```
        }
```



```
1
232
34543
4567654
```



```
2
43
654
```

Practical-5.12.7(Actual)

- Matrix of 4*9

I/J	1	2	3	4	5	6	7	8	9
1	*	*	*	*	*	*	*	*	*
2		*	*	*	*	*	*	*	
3			*	*	*	*	*		
4				*	*	*			
5					*				

For loop

1. Rows

2. Triangle-1 -space

3. Triangle-2-pattern

4. Triangle-3-pattern

Practical-5.12.7(Actual)

- Matrix of 4*9

1. Rows -for(i=0;i<5;i++)

2. Triangle-1 –space - for(j=0;j<=i; j++)

3. Triangle-2-pattern - for(k=5;k>i;k--)

4. Triangle-3-pattern – for(k=4;k>i;k--)

I/J	1	2	3	4	5	6	7	8	9
0	*	*	*	*	*	*	*	*	*
1		*	*	*	*	*	*	*	
2			*	*	*	*	*		
3				*	*	*			
4					*				

Practical-5.12.8(Actual)

for(i=1; i <=5; i++)

I/J	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5				
2	1	2	3	4					
3	1	2	3						
4	1	2							
5	1								

for (j=0;j<=i;j++)

```
1
12
123
1234
12345
```

for(k=5;k>=i; k--)

```
54321
5432
543
54
5
```

Practical-5.12.8(Actual)

1. Start
2. `for(i=1;i<5;i++)`
3. `c = 0`
4. `for(j=i;j<=5;j++)`
 - `c++;`
 - Print `c`End inner loop
5. End outer loop

I/J	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5				
2	1	2	3	4					
3	1	2	3						
4	1	2							
5	1								

Practical-5.13

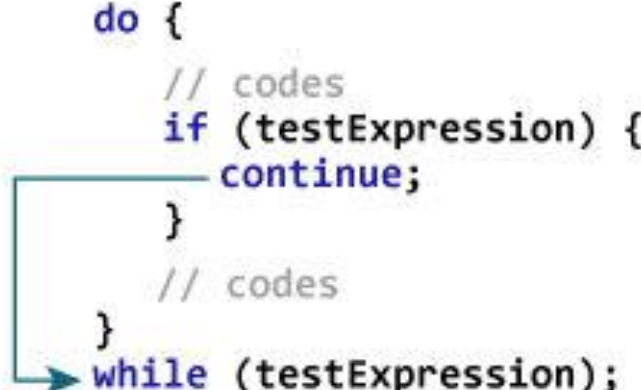
- Write a program to print multiplicative table like follow:
- Hint: use “continue;” statement in your loop to skip particular iteration.
- Use “break;” to limit only those multiplicative table up to which you want

Continue statement

```
while (testExpression) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```



```
do {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
} while (testExpression);
```



```
for (init; testExpression; update) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```



Continue statement


```
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==5)
        {
            continue;
        }
        printf("%d \t",i);
    }
    return 0;
}
```

OUTPUT:


1 2 3 4 6 7 8 9 10

Break statement


```
while (testExpression) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}
```



```
do {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
} while (testExpression);
```



```
for (init; testExpression; update) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}
```



Continue statement

```
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==5)
        {
            break;
        }
        printf("%d \t",i);
    }
    return 0;
}
```

OUTPUT:

1 2 3 4

Practical-5.13(1)

- Print multiplicative table like follow:

1. Start
2. Enter no
3. For $i=1$ to $i \leq 10$
4. Print table $n*i$
5. End

Practical-5.13(2)

- start
- Enter no
- Prinntf -Enter the skip point number
- Read no
- For loop
 - If i==skip_no value
 - Continue;
 - Print n*I
- End