→ for loop :-

- It is a conditional iterative control statement

Syntax :-

① ② ④

for(initialization ; condition check ; control variable operation)

{

S_1
S_2] ③ Body of loop
S_3

}

→ Statement above the for loop execute normally and sequentially.

Example :-

```
#include<stdio.h>
void main()
{
int i;
pf("Hello\n");
for(i=0;i<5;i++)
pf("Hai i=%d\n",i);
pf("Bye");
}
```

i. [T]

i=0 ; i<5 ; i++

0<5

Hai i=0 i++

1<5 Hai i=1 i++

2<5

Hai i=2 2++

⋮

5<5 [F]

o/p :-

Hello

Hai i=0

Hai i=1

Hai i=2

Hai i=3

Hai i=4

Bye i=5

Now consider the same program :-

```
for(i=0; i<5, i++); // Dummy loop
if ("Hai i = %d\n",i);
if ("Bye\n");
```

No statement But still for loop executes

o/p :- Hello
Hai i=5
Bye

Session 2?

Control statements :-
- for loop
- Multiplication Table using for loop.

Example:-

```
#include<stdio.h>
void main()
{
  int i, j;
printf("Hello\n");
for (i=0, j=10; i<j; i++, j--)
printf("Hai ... i=%d j=%d\n", i, j);

printf("Thanks\n");
}
```

any no. of variable can be initialized here.

→ In condition checking portion, no of expression can be written with a "comma", but the rightmost expression decides the entry into the loop. (Here comma acts as an operator.

when `for(i=0,j=10; j>=i, 0; i++, j--)`
This decide because '0' False the statement inside not done & no Inc/Dec

When: for(i=0, j=10; j>=i; i++, j--)

(1) → TRUE

and the loop is executed to infinity.

→ Example:-

```
Void main()
{
  printf("Hello\n");
for(int i=0; i<5; i++)
  printf("Hai i=%d\n", i);

printf("Bye i=%d", i);
```

Dynamic declaration
The scope of 'i=0' is only till the for loop

error

because 'i' is declared only for 'for' loop & cannot be used outside of loop.

☆ Note:- Initialization can be empty by (;) semi colon is must.

= Example:-

```
Void main()
{
  unsigned int i;
for(i=5; i>=0; i--)
{
  printf("Hai i=%d\n", i);
}
printf("Bye i=%d\n", i);
}
```

Due to this, this loop runs to infinity

5 >= 0 T
4 >= 0
3 >= 0
2 >= 0
1 >= 0
0 >= 0

after 0 that, as the range of unsigned int is from 0 to 65535

0 - - 65535

65535 >= 0 T

Loop continues

Note :- In a for loop, If there is no condition, the compiler will always consider 'TRUE'.

eg:- for (; ; ;)

Session 24 :-

→ Control statements
  → for
  → Multiplication Table using for.

(i) Rotate a loop 10 times :-

- for (i=0; i<=10 ; i++)
- for (i=1; i<=10 ; i++)
- for (i=1; i<11 ; i++)

- for (i=1 ; i<20; i=i+2)
- for (i=1 ; i<=19; i=i+2)

1 3 5 7 9 11 13 15 17 19 — 10 Times

- for (i=10; i>0 ; i--)
- for (i=10; i>=1 ; i--)

(ii) To rotate 5 times

for (i=20 ; i>10; i=i-2)

=

for (i=20; i>=12 ; i=i-2)

20 > 10
18 > 10
16 > 10
14 > 10
12 > 10

(iii) Rotate a loop for 10 times, half of it is +ve and half of it is -ve

-ve, +ve

for (i=-5; i<5; i++)
for (i=-5; i<=4; i++)

Multiplication table using for loop:-

```
{
int num,i;
pf("Enter the number \n");
sf("%d",&num);
for(i=1; i<=10; i++)//after complition of loop i=11
                    //so i=11
  pf("%d * %d = %d \n", num, i, num*i);
}
```

o/p:- The value of i=11 here.

for(i=1 i<=10; i++); Dummy loop

Now consider;

```
for(i=1;i<=10;pf("%d * %d = %d \n",num,i,num*i),pos--);
```

↑

→ It gives the normal o/p as exe. in the previous logic.

Printing the binary format of the number using for loop:

```
{
  int num, pos;
pf("Enter the number \n");
sf("%d",&num);
for(pos=31; pos>=0; pos--)
pf("%d", num>>pos&1);
pf("\n Thanks \n");
}
```

```
for(pos=31;pos>=0;pf("%d",num>>pos&1); pos--);
```

Write a program to count how many bits are set in a given number.

Any no. which is a power of 2, the no. of bits set is 1.

→
```
{
int num, pos, c;
pf("Enter the number\n");
sf("%d", &num);
for(pos=0, c=0; pos<=31; pos++)
{ // optional
if(num & 1<<pos) // Bit set 1, clear zero
c++;
} // optional
pf("c=%d\n", c);
```

To print both set and clear:

```
pf("set=%d\n clear=%d\n", c, 32-c);
```

To print the value and counting

```
for(pos=31; c=0; pos>=0; pos--)
{
if(num & 1<<pos) // set
{ pf("1");
c++
}
else
pf("0");
}
pf("\n set=%d clear=%d\n", c, 32-c);
}
```

Adding the digits of a given number

Here each digit has to be extracted % with 10, to do this

eg: take 127 % 10 = 7

127 % 10 = 7 // select digit

To delete the last number, divide by 10

num = num / 10

127 = 127 / 10 = 12

When the no. become '0' come out of the loop

Initially s = 0

r = num % 10

s = s + r

num = num / 10

Session 25

```
#include <stdio.h>
void main()
{
int num, r, sum;
printf("Enter the number\n");
scanf("%d", &num);

for(sum=0; num; num=num/10)
  {
   r = num % 10;
   sum = sum + r;
  }
printf("sum = %d\n", sum)
}
```

Condition check

This shows that the value of num should be TRUE i.e non zero or could be written as

num > 0 or num != 0

o/p:

Enter the number

1234

sum = 10

another logic:- int num, num1, sum, r;

→ for (sum=0, num1=num; num1; num1 = num1/10)
{
r = num1 % 10;
sum = sum + r;
}

Note:- In previous logic, the actual logic is lost. So here we introduce a new variable num1. to keep the number.

→ for (sum=0, num1=num; num; num1 = num1/10)
sum = sum + num1 % 10;

→ for (sum=0, num1=num; num1; sum = sum + num1 % 10,
num1 = num1/10);
pf("num = %d sum = %d \n", num, sum);

Reversing the digit of a given number:
eg:- 123 — 321

for (sum=0, num1=num; num1; num1 = num1/10)
sum = sum * 10 + num1 % 10;

☆ If 10 is multiplied it is for reversing
☆ If not then it is for adding.

⇒ { int num, num1, r, sum;
pf("Enter the num\n");
sf("%d", &num);

for (sum=0, num1=num; num1; num1 = num1/10)
sum = sum * 10 + num1 % 10;
pf("num = %d sum = %d \n", num, sum);
}

output logic:-

num = 123
Sum = 0 num1 = 123
Sum = 0*10 + 123 % 10
Sum = 0 + 3
num1 = 123/10 = 12

Sum = 3 num1 = 12
Sum = 3*10 + 12 % 10
= 30 + 2
Sum = 32
num1 = 12/10 = 1

Sum = 32 num1 = 1
Sum = 32*10 + 1 % 10
= 320 + 1 = 321

num1 = 1/10 = 0
False
Sum = 321

Write a program for reversing the bit of a program.
① How to check bit is set/clear.
② How to compliment a bit.

```
#include <stdio.h>
void main()
{
 int num, pos, i, j, m, n;
 printf("Enter the number\n");
 scanf("%d", &num);

 printf("Before rev num = %d\n", num);
 for(pos = 31; pos >= 0; pos--);
 printf("%d", num>>pos&1); //for printing binary of a number.
 printf("\n");

for(i=0, j=31; i<j; i++, j--)
{
 m = num>>i&1;
 n = num>>j&1;
```

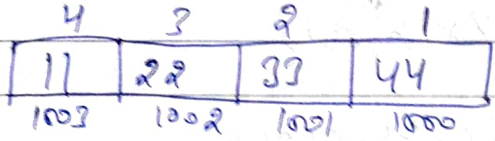
checking bit set or clear

```
if(m!=n)
{
num=num^1<<i; // The bits compliment.
num = num^1<<j;
}
}
pf("After reversing num = %d\n", num);
for(pos=31; pos>=0 ; pos--)
pf("%d", num>>pos&1);
pf("\n");
}
```

Session 26

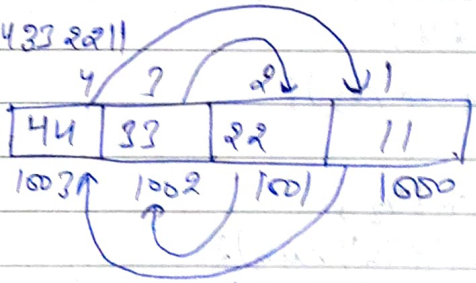
Converting little endian to big endian

Ex:- I/P:- 0x11223344 (Hexadecimal Representation)



Little endian

O/P:- 0x44332211



Big endian

```
Step (1):- r=num & 0XFF
Step (2):- r=r<<24  // r= 0x44
Step (3):- r1=num & 0000FF00
Step (4):- r1=r1<<8  // r= 00330000
Step (5):- r2= num & 0x00FF0000
Step (6):- r2=r2>>8  // 0x00022200
Step (7):- r3= num & 0xFF000000
Step (8):- r3= r3>>24 // r3 = 0x00000011
Step (9):- num= r|r1|r2|r3;
```

```
#include<stdio.h>
  void main()
{
unsigned int num=0x11223344
 int  r, r1, r2, r3;
pf("%x\n", num);

r=num&0x000000FF;
r=r<<24;  //LSB moves to MSB (1st to 4th byte)
pf("r=%x\n", r);

r1=num&0x0000FF00;
r1=r1<<8;
pf("r1=%x\n", r1);

r2=num&0x00FF0000;
r2=r2>>8;
pf("r2=%x\n", r2);

r3=num&0xFF000000;
r3=r3>>24;
pf("r3=%x\n", r3);

num=r|r1|r2|r3;
pf("num=%x\n", num);
   }
```

o/p:-
11223344
r = 44000000
r1 = 330000
r2 = 2200
r1 = 11
44332211

when an intel processor sends a message to any motorola processor, the byte ordering has to be changed from little endian to big endian

Prime Number

The number that has only 2 factor, 1 and itself : eg : 7, 13, 17, 19, 23 etc.

Rotate the loop untill the number itself

if % gives '0' for exact 2 times then it is a prime number

Write a program to find whether the given number is prime or not.

→
```
{
 int num, c, i;
 pf("Enter the number\n");
 sf("%d", &num);

for(i=1, c=0; i<=num; i++)
 {
  if(num % i == 0)
    c++;
 }
 if(c==2)
 pf("Prime number\n");
 else
 pf("Composite or not prime number\n");
       }
```

→ Alter logic for Prime

```
for(i=2, c=0; i<num; i++)
 { if(num % i == 0)
     c++;
 }
```

```
if (c==0)          // because for loop starts from 2 and end before the itself. (in b/w no factor to 'prime'.
pf("prime\n");
else
pf("Not prime\n");
    }
```

→ Alter logic :-

```
for(i=2; i<num; i++)
  if (num % i ==0)
    break;

if (num==i)
pf("Prime\n");
else
pf("Not prime\n");
    }
```

Session - 29

→ Control statements
  → nested for loop
  → while loop
  → Factorial of a given number.

Perfect numbers

A +ve integer that is equal to the sum of its proper divisors.

Ex:- 6 (1 + 2 + 3) = 6.

Program of perfect number.

```
{
    int sum=0, num, i;
for (i=1; i<num; i++)
  {
  if(num % i ==0)
  sum = sum + i;
      }
```

pf( )
sf( )

```
if (sum == num)
  pf("Perfect number \n");
else
pf("Not perfect \n");
        }
```

Armstrong number:

Is a number that is equal to the sum of cubs of its digits.
Eg: 0, 1, 153, 370, 371, 407 etc.

```
{ int sum, num, num1, r;
→ for (sum=0; num=num1; num1=num1/10)
   {
    r = num1%10;          //Extract every digit
    sum = sum + r*r*r;    // Cube the digit then
   }                          add these.
   if (sum == num)
   pf("Armstrong number \n");
   else
   pf("Not armstrong number \n");
     }
```

Factorial of a given number.

```
{  int num, fact, i;
   pf("Enter the number \n");
   sf("%d", &num);          //f=0 X because 0*0 wrong with anyone =0

for (i=1, f=1; i<=num; f=f*i, i++);
   pf("The factorial of %d = %d\n", num, f);
        }
```

Nested loop
(loop within loop)

To perform one task, if we want to perform that task repeatedly, we need to go for nested loop.

Note:- While nesting the loops, outer and inner loop control variable should be different.

eg:-

```
void main()
{
  int i, j;
  for(i=0; i<4; i++)        // outer loop 4 times 0,1,2,3
  {
    for(j=0; j<3; j++)      // inner loop 3 times 0,1,2
    {
      printf("%d,%d", i, j);
      printf("Hai\n");      // 0,0 Hai  0,1 Hai  0,2 Hai
    }                       // 1,0 Hai  1,1 Hai  1,2 Hai
    printf("-----");        // 2,0 Hai  2,1 Hai  2,2 Hai
  }                         // 3,0 Hai  3,1 Hai  3,2 Hai
  printf("Bye\n");          // -----
}                           // Bye
```

Session 28

Note:- If loops are not nested or dependent same control variable can be used.

Examples of Nested loops:-

```
void main()
{
  int i, j;
  for(i=0; i<5; i++)
```

{
for(j=0; j<5; j++)
pf("%d", j);
pf("\n"); // outer loop's control variable can be accessed from the inner loop
}
}

i	j
0	0,1,2,3,4
1	0 1 2 3 4
2	0 1 2 3 4
3	0 1 2 3 4
4	0 1 2 3 4

o/p:
00000
11111
22222
33333
44444

→ #include<stdio.h>
void main()
{ int i,j;
for(i=0; i<5; i++)
{
for(j=0; j<5; j++)
pf("%d_", j); // space
pf("\n");
}
}

o/p:-
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4

→ for(i=0; i<5; i++)
{
for(j=0; j<=i; j++)
pf("%d_", j);
pf("\n");
}

// i=0 inner loop rotate 1 time
// i=1 inner loop rotate 2 time
// i=2 inner loop rotate 3 time
so on

o/p:-
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4

{
→ char int i,j;
char ch;
for(i=0, ch='a'; i<5; i++, ch++)
{
for(j=0; j<=i; j++)
pf("%c_", ch);
pf("\n");
}
}

o/p:-
a
b b
c c c
d d d d
e e e e e

→
```
for(i=0 ; i<5 ; i++)
{
  for(j=0, ch='a' ; j<=i ; j++, ch++)
    Pf("%c_", ch);
  Pf("\n");
}
```

o/p:-

a
a b
a b c
a b c d
a b c d e

Pattern printing

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

o/p

*
* *
* * *
* * * *
* * * * *

#

* * * *
* * *
* *
*

```
for(i=0; i<4; i++)
  {
for( j=0 ; j<4-i , j++)
  Pf("* ");
Pf("\n");
  }
```

i	j	* times
0	4	
1	3	//
2	2	//
3	1	//

0 j<4
1 j<3
2 j<2
3 j<1
4 j<0 X

# void main()
{
int i, j, n;
pf("Enter n\n");
sf("%d", &n);

for(i=0; i<n; i++) → Lower triangle
{
for(j=0; j<=n-1; j++)
pf("* ");
pf("\n");
}
}

Session 29 9/-7-2-

→ C S
→ Prime number within range

Prime no.s from 50 to 100 (using nested for loop)
main logic

for(num=50; num<=100; num++)
{
for(i=2; i<num; i++)
if(num % i == 0)
break;
if(num == i)
pf("%d", num);
}

Print first 15 prime number starting from 50.

#include<stdio.h>
void main()

```
{
int num, i, c;
for(num=50, c=0; c<15; num++)
{
  for(i=2; i<num; i++)
    if (num%i==0)
      break;
  if (num==i)
  {
    c++;
    pf("%d", num);
  }
  pf("\n");
}
```

Print palindrome nos. in b/w 100 to 200.

```
void main()
{
  int num, num1, sum;
  for(num=100; num<=200; num++)
  {
    for(num1=num, sum=0; num1; num1=num1/10)
      sum=sum*10+num%10; // palindrom
  if (sum==num)
    pf("%d", num);
  }
  pf("\n");
}
```

logic for reversing number

Pattern printing:-

①

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

"\n" outer loop	* inner loop
0	1 time
1	2 times
2	3 times
3	4 times
4	5 times

```
{
int i,j;
for(i=0;i<5;i++)
 {
  for(j=0;j<=i;j++)
   pf("*");
 pf("\n");
   }  }
```

Now in more generic way

```
{ int i,j,num;
 pf("Enter a number \n");
 sf("%d",&num);
  for(i=0; i<num; i++)
 {
  for(j=0 ; j<=num; j++)
   pf("*");
   pf("\n");
   }  }
```

(2)

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

i=0 j=5
i=0 j<5-i

(i) outer loop	(j) inner
0	5 times
1	4 times
2	3 times
3	2 times
4	1 time

```
void main()
{
 int i,j;
for(i=0;i<5;i++)
   {
    for(j=0;j<5-i;j++)
pf("*");
pf("\n");
 }
   }
```

③ space _ *

```
____*  *
___* * *
__* * * *
_* * * * *
* * * * *
```

i	space	*
0	4	1
1	3	2
2	2	3
3	1	4
4	0	5

```
{
 int i, j, k;
for(i=0; i<5; i++)
 {
  for(j=0; j=4-i; j++)
   pf(" ");

 for(k=0; k<=i; k++)
   pf("*");
   pf("\n");
   }
}
```

④

```
* * * * *
_* * * *
__* * *
___* *
____*
```

outer i	space	star *
0	0	5
1	1	4
2	2	3
3	3	2
4	4	1

⑤

```
*__ *
__* *
_* * *
* * * *
_* * *
__* *
___*
```

```
for(i=0; i<5; i++)
{ for(j=0; j<4-i; j++)
   pf(" ");
 for(k=0; k<i; k++)
   pf("*");
   pf("\n");
}
for(i=0; i<3; i++)
 { for(j=0; j<=i; j++)
   pf(" ");
 for(k=0; k<3-i; k++)
   pf("*");
   pf("\n"); }
```

Date 21-07-20

Session - 3.

→ Control statements
  → while loop
  → do while loop.

While loop:

→ while is an iterative control statement.

Syntax:

```
while (condition)
{
  S1 ;
  S2 ;   } Statement
  S3 ;
}
```

→ while loop has only 2 portions:-
1. condition checking
2. body of loop.

while loop is faster when compared to for loop, as while loop contains only two portions.

⇒ In a while loop, if the condition is left empty, then it is considered as a Syntactical error.

But in a for loop, if the condition is left empty, then it goes to infinity.

→ while (exp) if brackets are not given,
S1 S1 is a part of the while loop
S2
S3

```
while (exp);   // This is a dummy while loop
{
S1;
S2;
S3;
```
} 3 statements do not belong to the while loop.

```
while (1)
{
   ≡
}
```
→ this will execute continuously (execute forever)

for (① ; ② ; ④)	① Initialization while (②)
{	{
③	③
}	④
	}

- 1. Initialization
- 2. Condition check
- 3. body of loop
- 4. Ctrl variable operation

Note: If the no. of times, the loop should rotate is known, go for 'for' loop.

Example:

```
While ((ch = get_char_data()) != '$')
{
   ≡
}
```
↳ This expression receives data until it receives '$' symbol

Here, the no. of rotation is NOT known. Thus, a while loop is used here.

Multiplication table program using while loop.

```
{
int num, i;
pf("Enter the number");
```

```
pf("%d",sum);
i=1;          // initialisation
while (i<=10) // Condition checking
{
pf("%d * %d = %d\n",num,i,num*i); //body
i++;
}
pf("Thankyou\n");
}
```

→ Example:-

```
{
int i=0;
pf("Hello\n");
while (i<4)
{
pf("Hai i=%d\n",i);
i++;
}
pf("Thankyou i=%d\n",i);
}
```

0<4
1<4
2<4
3<4
4<4 X → comes out of while loop

O/P
Hello
Hai i=0
Hai i=1
Hai i=2
Hai i=3
Thankyou i=4

```
i=1;
Now:
while (i++<4)
pf("Hai i=%d\n",i);
pf("Bye i=%d\n",i);
```

i=1
1<4
2<4
3<4

O/P
Hai i=1
Hai i=2
Hai i=3
Bye i=4

```
i=0;
while (i++)
pf("Hai i=%d\n",i); // Not printed.
pf("Bye i=%d\n",i);
```

O/P = Bye i=1

```
i=0;
while (i++);
pf("Hai i=%d",i);
pf("Bye i=%d",i);
```

Hai i=1
Bye i=1

→ while (i++, i);
printf("Hai");
printf("Bye");

Due to ';' this while is treated as dummy while & the digit 'i' make always true, so the value tends to infinity.

Do-While loop:-

→ This loop will executes minimum 1 time.
→ while is an entry controlled loop whereas, do while is an exit controlled loop.
→ while executes minimum zero.

Syntax:-

```
do
{
≡≡≡
} while(exp);
```

```
do
statement;
while (condition);
```

Example:-

```
{
int i=0
printf("Hello\n");
do
printf("Hai...\n");
while(1)
printf("Bye\n");
}
```

O/P

Hello
Hai
Bye