```
        Printf " (great ");
        exit(1);
    }

sos:
        Printf ("To err is human\n");
}.
```

Any no. of goto.

exit(1) → lib fn. <stdlib.h>

Looping statements

→ actions over & over → with variations each time.

(Looping)

→ set of instrs repeatedly.

→ specified no: of times | or till a condition is reached.

while loop

→ repeat till the cdtn remains true.

Syntax:

```
while (condition)
        statement;

while (condition)
{
    stmt1;
    stm22;
}
```

```
main()
{
    int count = 1
    while (count <= 5)
    {
        Printf ("%d", count);
        count = count + 1;
    }
}
```

executed till the cdtn is false

o/p:-

1
2
3
4
5

loop counter / index Variable.

int / float

++ / --

trap 1:-

```
int i=1
while (i<=10)
    Printf ("%d\n", i);
```

→ loop counter may be a float.

trap 2:-

```
while (i<=10);
{
    Printf ("%d\n", i);
    i=i+1;
};
```

eplx: while (i>=10 && j<=15).

o/p:

```
main()
{
    int i=0;
    while (i++<10)
        Printf("%d\n", i);
}
```

1
2
3
:
10.

Prog:-
write a c program to find the factorial of a num.

```
main()
{   int fact, num, i;
    fact = i=1;
    Printf(" Enter a number");
    scanf ("%d, &num);
    while (i<=num)
    {
        fact = fact * i;
        i++;
    }
    Printf ("fact", fact);
}
```

H/w.
write a c program to find the power of another number.

O/p:-

```
while ('a' < 'b')
{
    Printf ("malayalam is a palindrome")
}
```

Printed indefinite

2)
```
while (i=10)
{
    Printf("%d\n", i);
    i=i+1;
}
```
Prints 10 indefinite

3)
```
float x=1.1;
while (x==1.1)
{
    Printf("%f\n", x);
    x = x-0.1;
}
```
1.000110011001100110

No o/p. float variable is compared to double constant

4) do-while

```
do
{
}while (cdtn);
```

```
float x=0.1;
if (x==0.1)
    Printf("IF");
else if (x==0.1f)
    Printf("ELSE IF");
else
    Printf("ELSE");
```

dif: while → entry condition
do-while → exit condition.
↓
executes atleast once, even if the cdtn fails for the first time.

(ex):
```
while (4<1)
{
    Printf("  ");
}
```

```
do
{
    Print.....
}
while (4<1);
```

Program:

Sum of the digits.

```
int sum=0, num, b;
Printf(" enter num");
scanf(" %d ", &num);
do
{ sum = sum +
    b= n % 10;
    n= n/10;
    ~~sum = sum + b~~;
}
while (num>0);

Printf(" %d", sum)
```

567
sum = 7
n = 56
7 + 6 = 13
n = 5
sum = 18

fibonacci Series

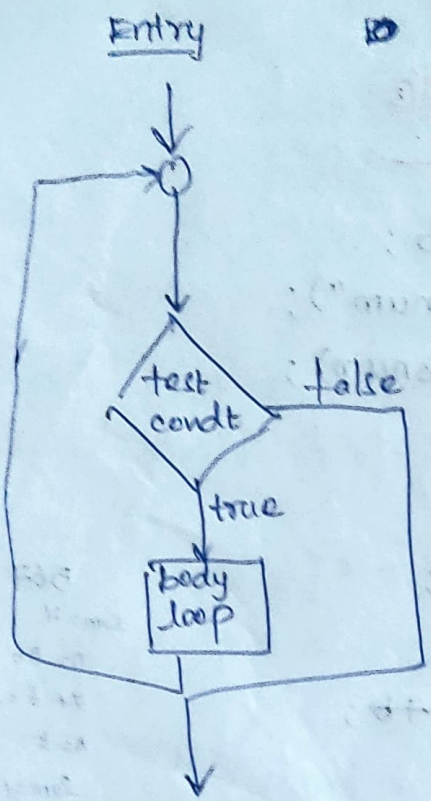
```
int num, i = 3, f1=0, f2=1;
Printf (" Enter the limit");
scanf (" %d" &num)
```

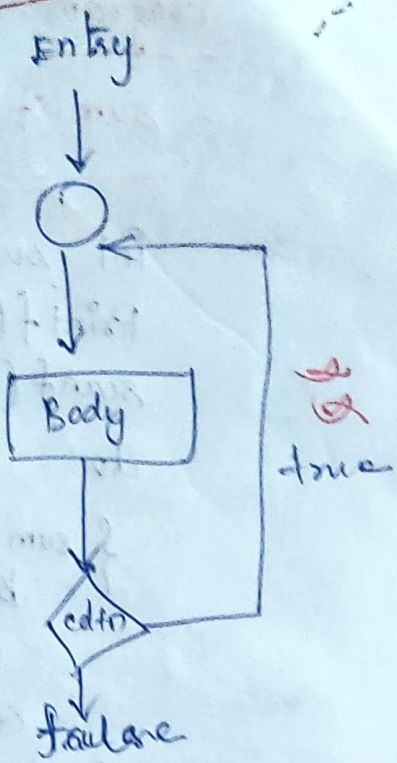
~~f=0;~~
~~f1=1;~~
~~f=f1;~~

Printf first 2 terms
f1, f2

```
do
{
    next = f1 + f2
    f1 = f2
    f2 = nextterm
    ~~i++;~~
    ~~Printf(" %d\n", f);~~
    Print = nexttterm
    ~~f=f1+f2;~~ }while(i<=n);
```

1 error -> to be said





Dif

~~Entry condition~~

cdtn checked by the loop body is executed

false, body is not executed even once

Pre-test / entry controlled loop

Semicolon :- not req

~~exit cdtn~~

cdtn checked after executing the body

at least once.

Post-test / exit control loop

Semi-colon req.

Bunch-3

For-loop:- entry-controlled loop

i) Initial value
ii) testing the loop counter variable
iii) Increment

```
for ( initialization ; condition ; increment )
{
    body;
}
```

body; → done only once. → (one stmt no braces)

ex

```
for (int i=1; i<=5; i++)
{
    Printf("%d\n", i);
}
```

1st

o/p.
1
2
3
4
5

no declaration, only initialization.

traps:-

3rd
```
for(i=1; i<=10; Printf("%d ", i++));
```
↳ 12345678910

1st
```
for ( i=1; i<=5; i++);
{
    Printf("%d");
}
```
→ o/p → 6

→ treat as separate block.

trap1:

```
for(j=1000; j>0; j=j-1);
```
↓ time delay for loops.

```
for ( i=1; i<=10; )
{
    Printf ("%d\n", i);
    i++;
}
```
→ but ✓ semi-colon necessary.

while () → error ← for (;;) → infinite loop.

trap 2:

```
int i = 1;
for (; i<=10; i=i+1)
        Print("%d\n", i);
```

initialization is done outside, but semi-colon necessary.

```
for (; i<=10;)
```

↓ 2 semi-colon necessary.

```
for (i=0; i++<10;)
```

(++i/i--)

↓ both in same segment.

Nested for loops

```
int r, c, Sum;
for(r=1; r<=3; r++)
{
    for(c=1; c<=2; c++)
    {
        Sum = r+c;
        Printf("%d %d %d", r, c,
                            Sum);
    }
}
```

O/P

r=1 c=1 Sum=2
r=1, c=2 Sum=3
r=2, c=1, Sum=3
r=2, c=2, Sum=4

r=3, c=1, sum=4
r=3, c=2, sum=5

2 for nested
2 while nested
for and while can be nested.

cdtll:- for(i=1, j=2; j<=10; j++) ✓
for(i=1, j=2; j<=10
&& i<=15; j++, i=i+2)

Prog:-
Largest of n numbers, using for loops:-

```
int n, i, max=0, c;
Printf("\nEnter the no: ");
Scanf("%d", &n);
for(i=1; i<=n; i++)
{
    Printf("Enter no: ");
    &canf("%d", &c);
    if(c > max)
    {
        max = c;
    }
}
Printf("%d", max);
```

break statement:-

↓ jump out of the loop, without waiting for the condition.

break → { for, while, do-while } → control passes to the next statement outside the loop.

for nested loop, it comes out of the inner loop only.

Prime-number:-

```
i=2;
while (i<=num-1)
{
    if(num%i==0)
    {
        Printf("Not a prime\");
        break;
    }
    i++
}
if(i==num)
    Printf("prime number\n");
}.
```

continue

- skipping a part of loop
- skipping current iteration.

→ control jumps to the beginning of the loop.

ex)

```
for(i=1; i<=5; i++)
{
    if (i==3)
        continue;
    Printf ("%d", i);
}
```

O/P

1 2 4 5.

	For	while	do-while
continue	goes to updation	goes to beg. (cdtn)	goes to condition
break	" inner for loop.	comes of the loop	"
goto	goes to the label anywhere!	→	may be inside/ outside the loop

Printing patterns

1)

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

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
int rows = 5
    for(i=0 ; i<=rows; i++)
    {
      for( int j=0 ; j<=i; j++)
      {
          Print ("*");  j
      }                  % d-j
                         % c- j+'A'
      Print("\n");
    }
```

2)

```
        1
      1 2  3
    1 2 3  4  5
  1 2 3 4  5  6  7
1 2 3 4 5  6  7  8 9
```

Odd loop

```
do
{
    Printf ("num");
    &canf ("%d", &num);
    Print ("%d", num * num);
    Print ("Want to enter
                another number?\n");
    &canf (" %c", &another);
}while (another == 'y');
```

H/w: write a C program to generate all combinations of 1,2 and 3 using for loop.

2)

1

2 3

4 5 6

7 8 9 10.

Chapter 2

Arrays

collection of similar data types

Array of 'char's → String

Array of int/float → array.

ordered sequence of homogenous values.

Syntax

decl: datatype name [size];

↓

dimension

ex) int marks [30];

marks[0], marks[1] .... marks[29].

↓ Starts with zero. ↘ subscript.

ex)

```
int avg, sum=0
int i;
int marks [30];
for(i=0; i<=29; i++)
{
    Printf ("enter marks");
    Scanf ("%d", &marks[i]);
                    → store
```

Invalid

double x[];

int N;

double x[N]; ✓ allowed