Loops

Lecture 6

Loops

- Loops used for the program to be executed repeatedly while expression is true.
- When the expression becomes false ,the loop terminates and the control passes on to the statement following the loop.
- Consists of two segments:

    -control statements

    -body of the loop

Kinds of Loops:

- for
- while
- do-while

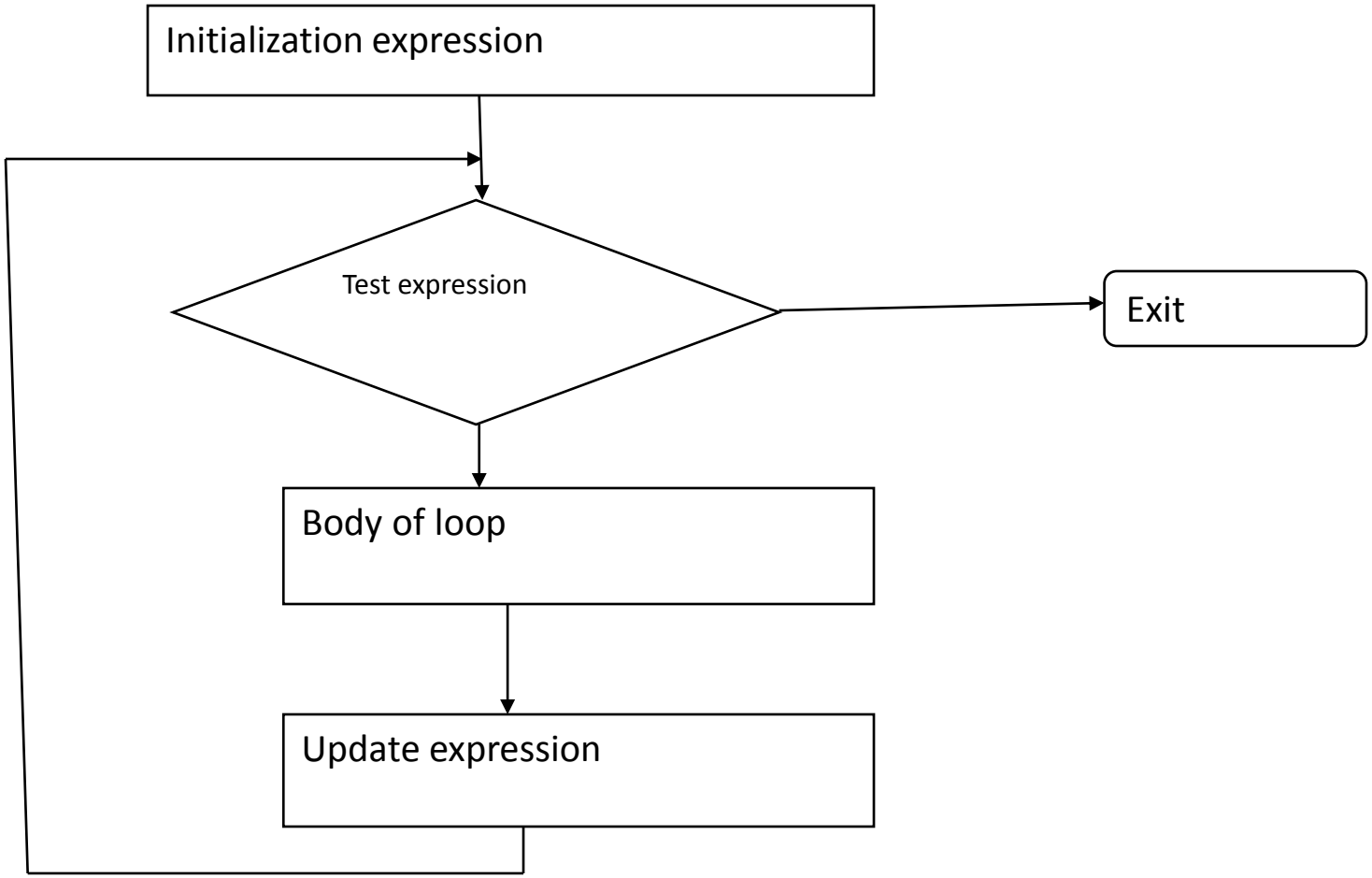
for loop

- Is useful while executing a statement a number of times.

syntax

for(initial expression; test expression; update expression)

statement/compound statement


Initialization expression
Test expression
Exit
Body of loop
Update expression

```
//A program to display the first 10 multiples 5 on
   a single line
#include<stdio.h>
 main()
{
 int i;
```

```
    for(i=1;i<=10;i++)
        printf("%d",5*i);
}
```

-for keyword is followed by the components enclosed within round braces(),separated by semicolons.

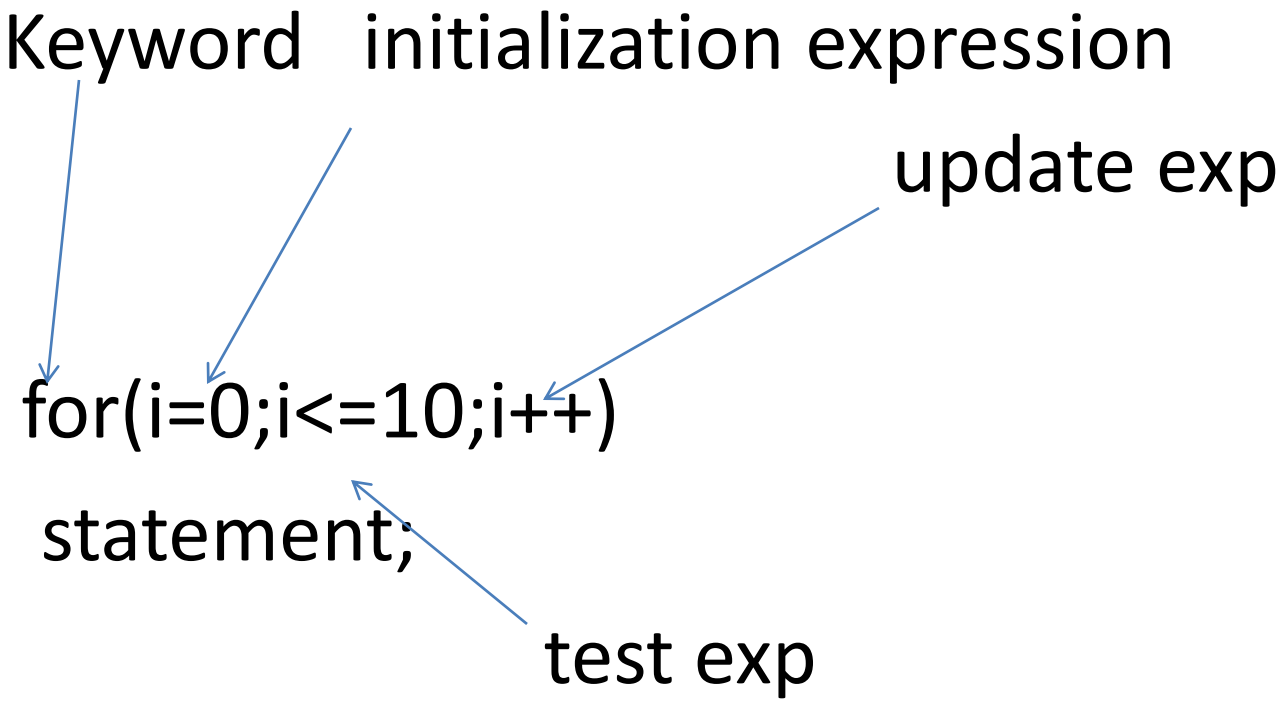
i=1

i<=10

i++

- First component i=1 is executed only once prior to the statements with in the for loop called initialization expression.
- Second component i<=10 is evaluated once before every execution of the statements with in the while loop called test expression.
- If the expression is true ,the statements with in the loop executes.

- If it is false ,the loop terminates and the control of execution is transferred to the statements following the for loop .
- The third component i++ is executed once after every execution of the statements with in the loop.
- The expression increments value of i by 1 called update expression.


Keyword
initialization expression
update exp
for(i=0;i<=10;i++)
statement;
test exp

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

  {

statement1;

statement 2;

  }

for(i=1;i<=10;i++) with no body

```
/* program to calculates sum of even*/
#include<stdio.h>
 main()
{
 int  i;
 int sum=0;
 for(i=2;i<=30;i++)
{
```

```
 sum+=i;
}
 printf("sum of first 15 even numbers=%d",sum);
}
```

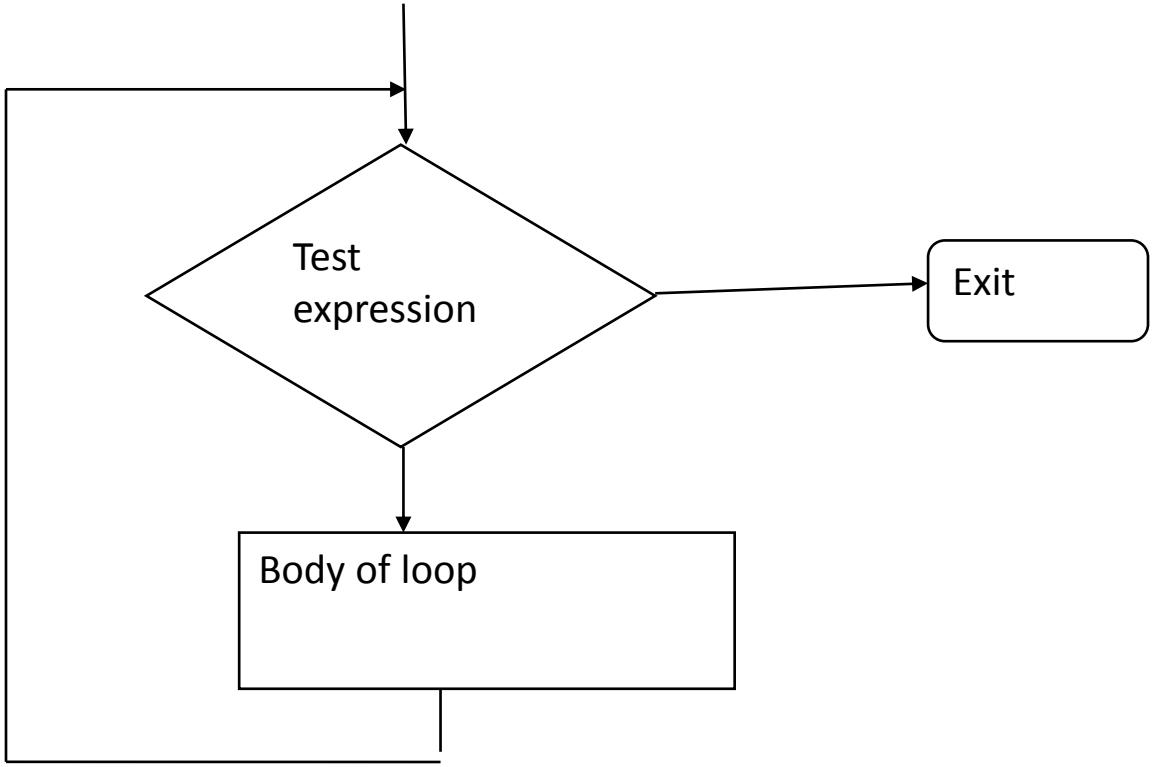
while loop

- while loop is often used when the numbers of times the loop is to be executed is not in advance.

syntax

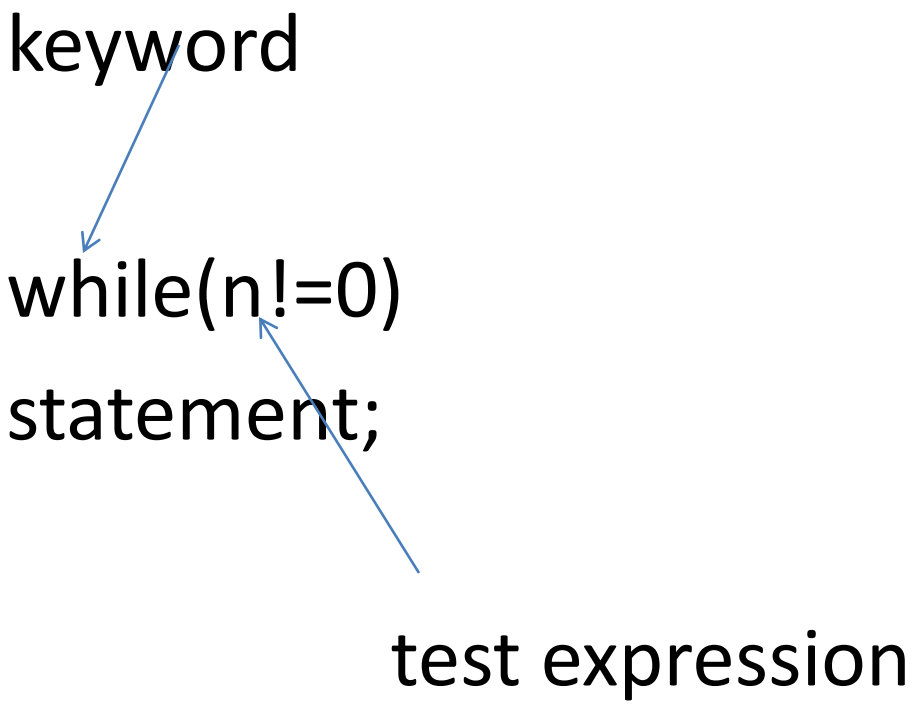
for single statement:

```
while(test expression)
statement;
```


Test
expression
Exit
Body of loop

```
 while(test expression)
{
 statement;
 statement;
……….

}
```


keyword
while(n!=0)
statement;
test expression

keyword

```
while(n!=0)
statement;
……
statement;
```

test expression

do-while loop

- while loop is top tested, it evaluates the condition before executing any statements in its body.
- In do-while evaluates the condition after the execution of the statements
- do-while loop are executed at least once.
- do-while loop is button tested.

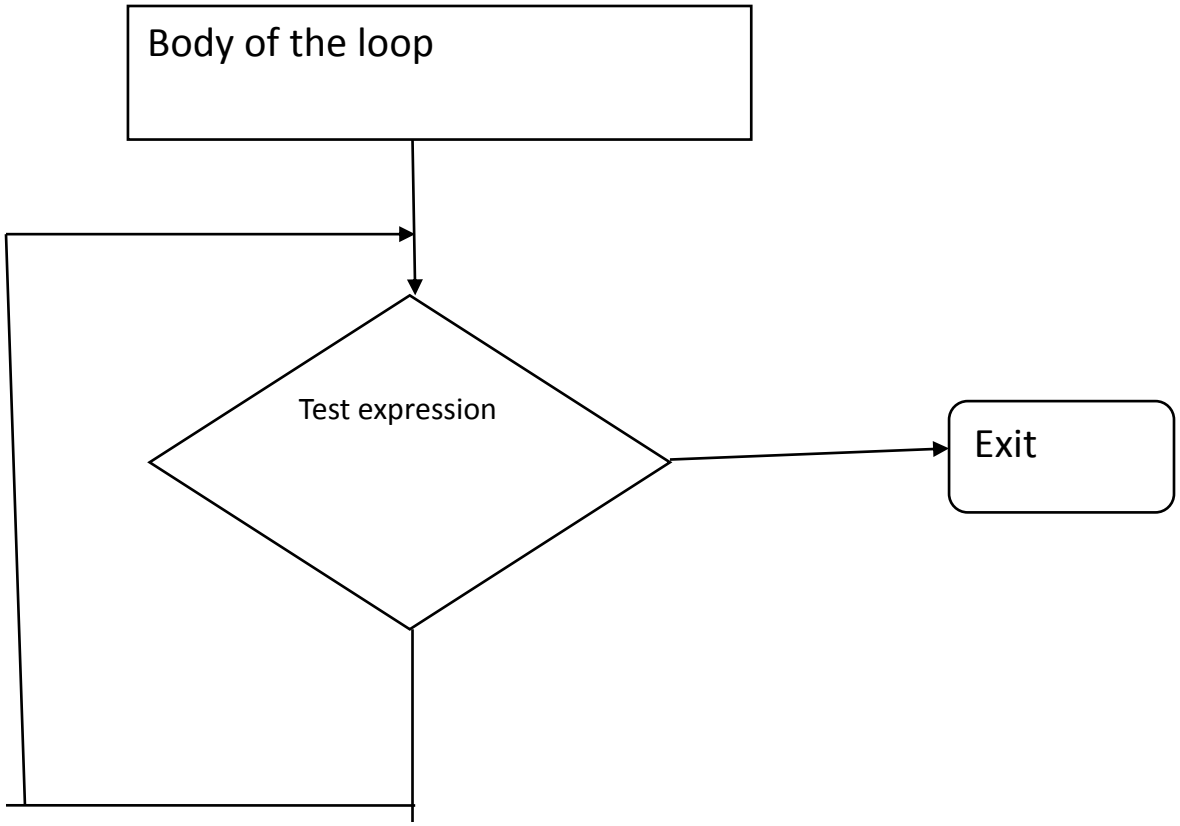
syntax

do

statement;

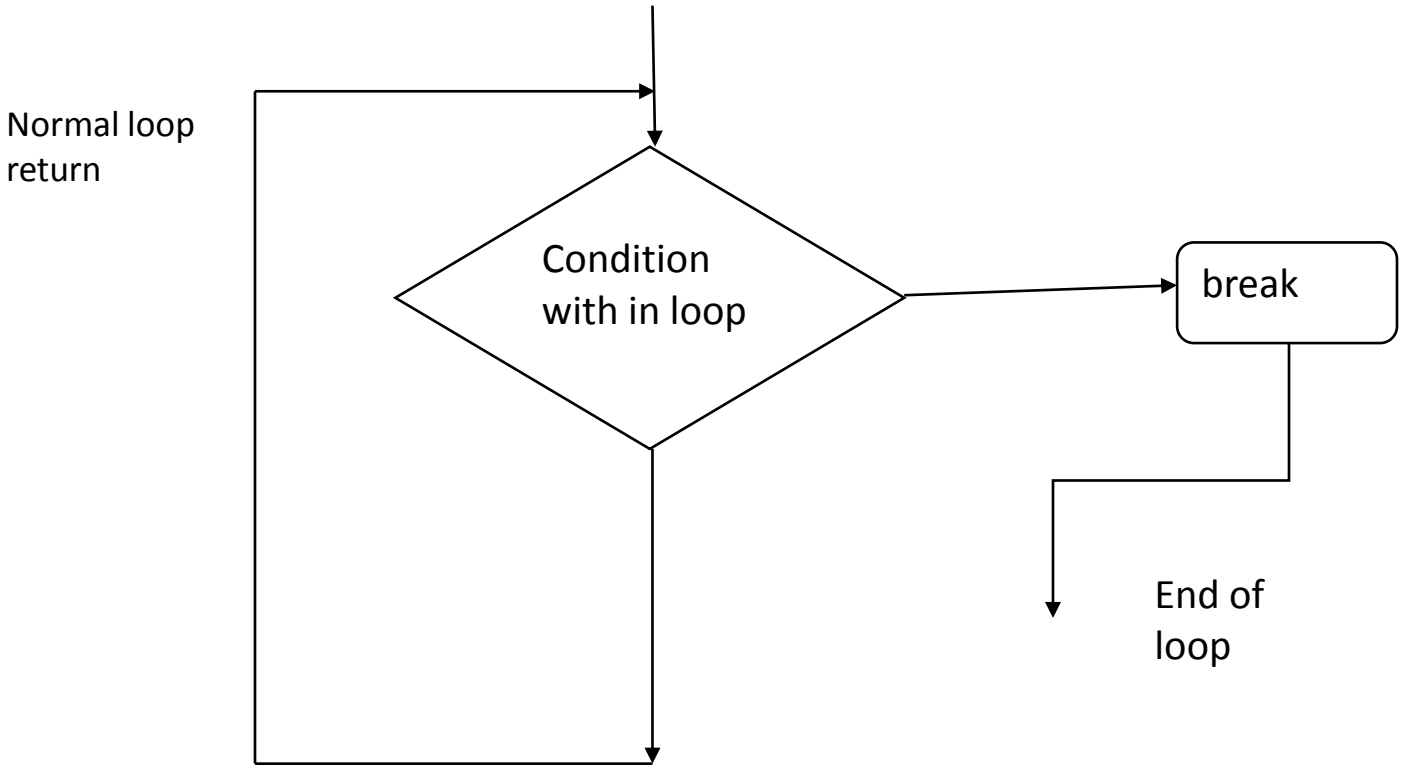
while(test expression);

```
 do
{
 statement;
 statements;
……….……
}
 while(test expression);
```


Body of the loop
Test expression
Exit

break statement

- terminates the execution of the loop and the control is transferred to the statement immediately following the loop.

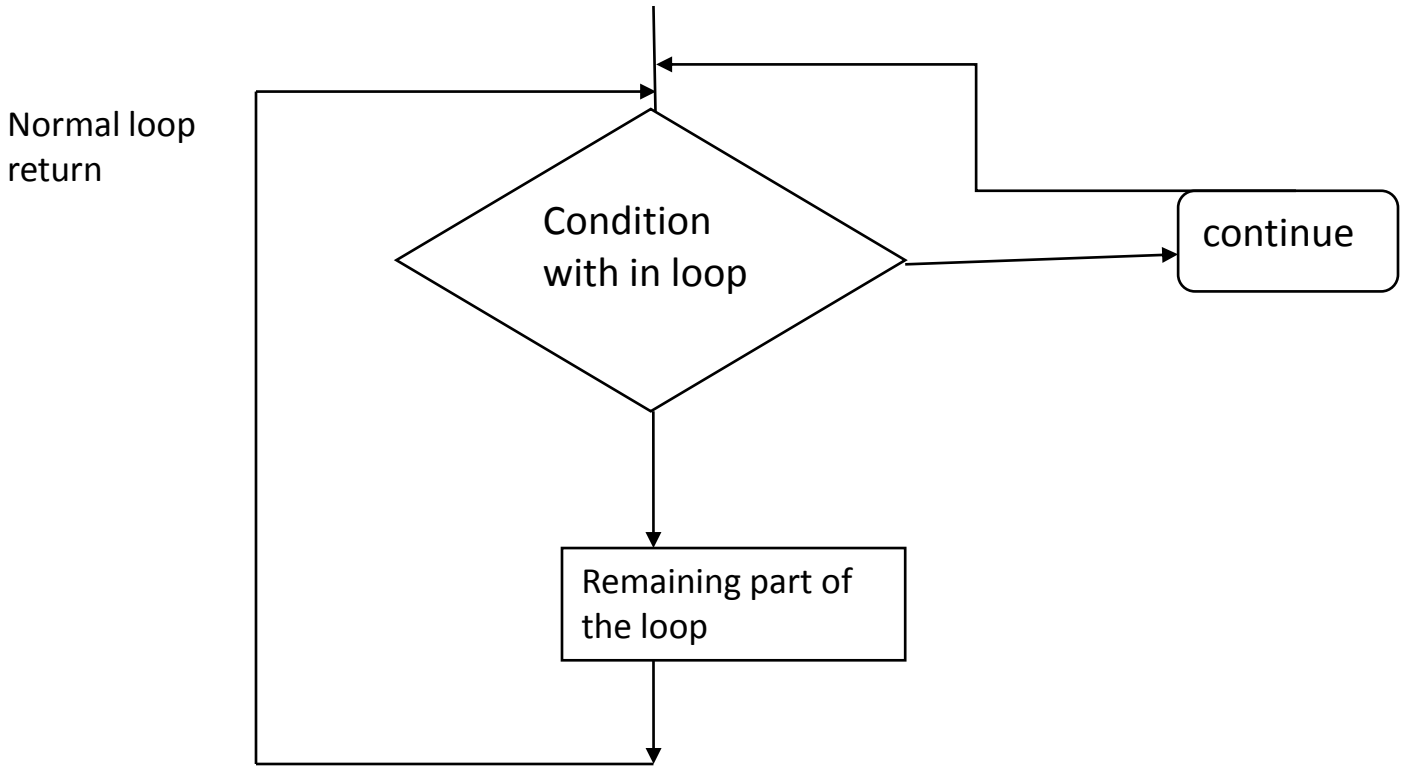

Normal loop
return
Condition
with in loop
break
End of
loop

continue statement

- Is used to bypass the remainder of the current pass through a loop.
- The loop does not terminates when the continue statements is encountered.

syntax

continue;


Normal loop
return
Condition
with in loop
continue
Remaining part of
the loop

Program to calculate avg of marks

```
#include<stdio.h>
main()
{
int n,count=1;
float x, avg, sum=0;
printf(“how many number:”);
scanf(“%d”,&n);
```

```
 while(count<=n)
{
 printf(“x=”);
 scanf(“%d”,&x);
 sum+=x;
 count++;
 }
```

```
  avg=sum/n;
 printf(“the avg is %f\n”,avg);
}
```