CONTROL STRUCTURES (REPETITION STRUCTURE)

JUMP STATEMENTS

Outline

- Repetition structure/Control Loop Statements
  - for statement
  - while statement
  - do-while statement
- Jump Statements
  - break
  - continue
  - goto
  - return

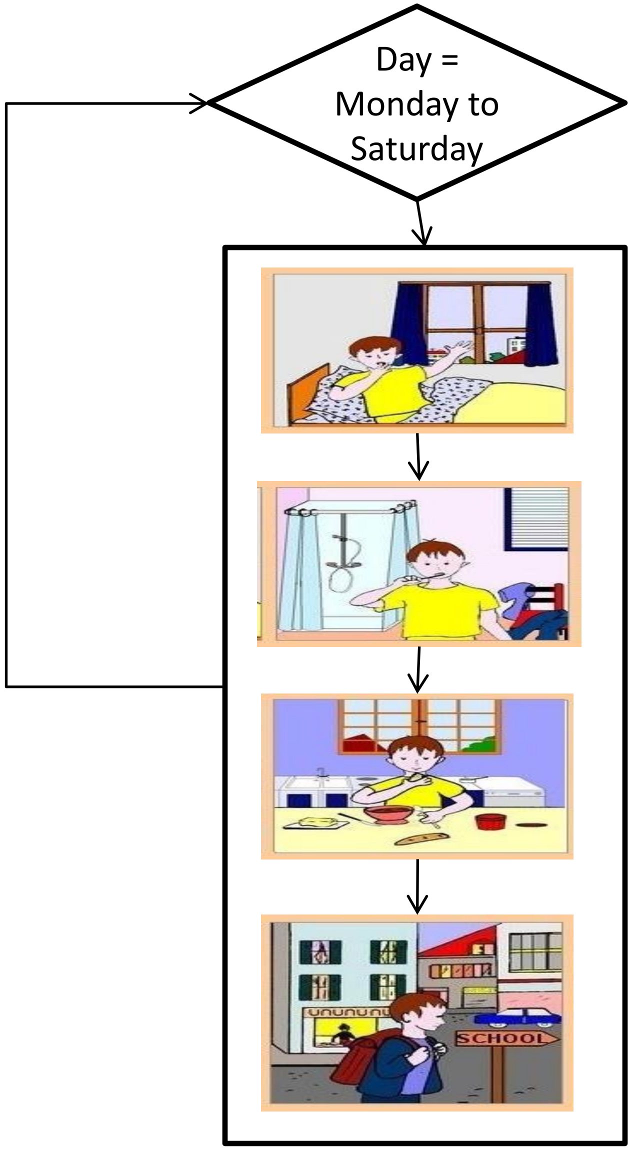
Difference between if-else and switch

Basis for Comparison	if-else	switch
Basic	Which statement will be executed depend upon the output of the expression inside if statement.	Which statement will be executed is decided by user.
Expression	if-else statement uses multiple statement for multiple choices.	switch statement uses single expression for multiple choices.
Testing	if-else statement test for equality as well as for logical expression.	switch statement test only for equality.
Evaluation	if statement evaluates integer, character, pointer or floating-point type or boolean type.	switch statement evaluates only character or integer value.
Sequence of Execution	Either if statement will be executed or else statement is executed.	switch statement execute one case after another till a break statement is appeared or the end of switch statement is reached.
Default Execution	If the condition inside if statements is false, then by default the else statement is executed if created.	If the condition inside switch statements does not match with any of cases, for that instance the default statements is executed if created.
Editing	It is difficult to edit the if-else statement, if the nested if-else statement is used.	It is easy to edit switch cases as, they are recognized easily.

Conclusion:

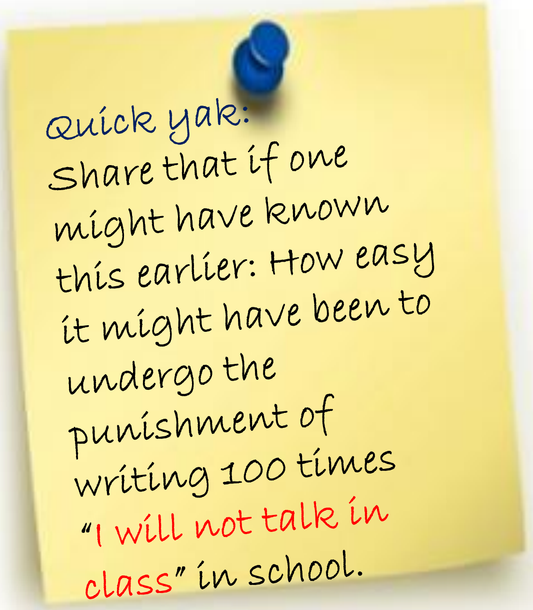
The switch statement is easy to edit as it has created the separate cases for different statements whereas, in nested if-else statements it become difficult to identify the statements to be edited.

Repetition(Going to School)



Repetition Statement

- A **repetition statement** allows you to specify that an action is to be repeated while some condition remains true.



Quick yak:
Share that if one might have known this earlier: How easy it might have been to undergo the punishment of writing 100 times "I will not talk in class" in school.

Looping (repetition)

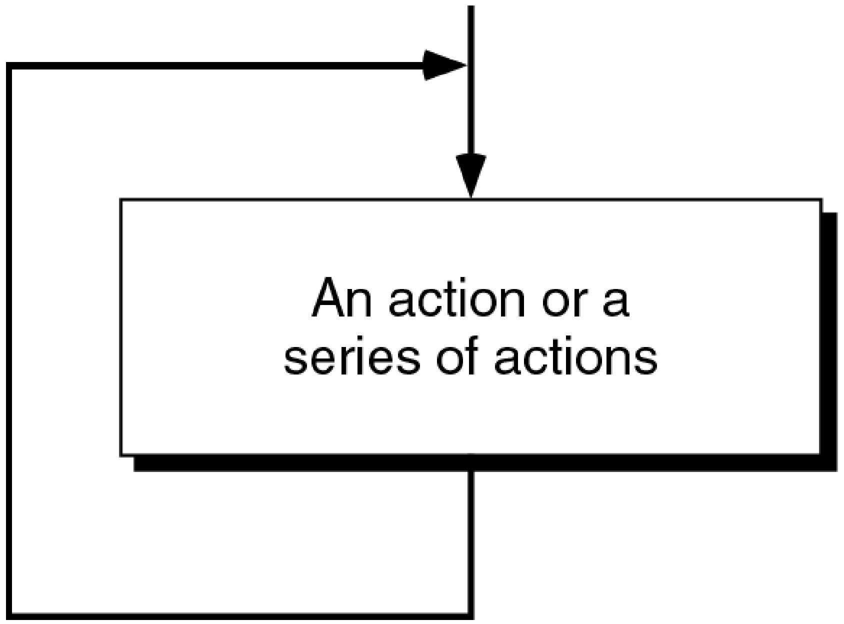
- *What if we want to display hello 500 times?*
  - Should we write 500 printf statements or equivalent ?

➢ Obviously not.

- It means that we need some programming facility to repeat certain works.
- Such facility is available in form of ***looping statements.***

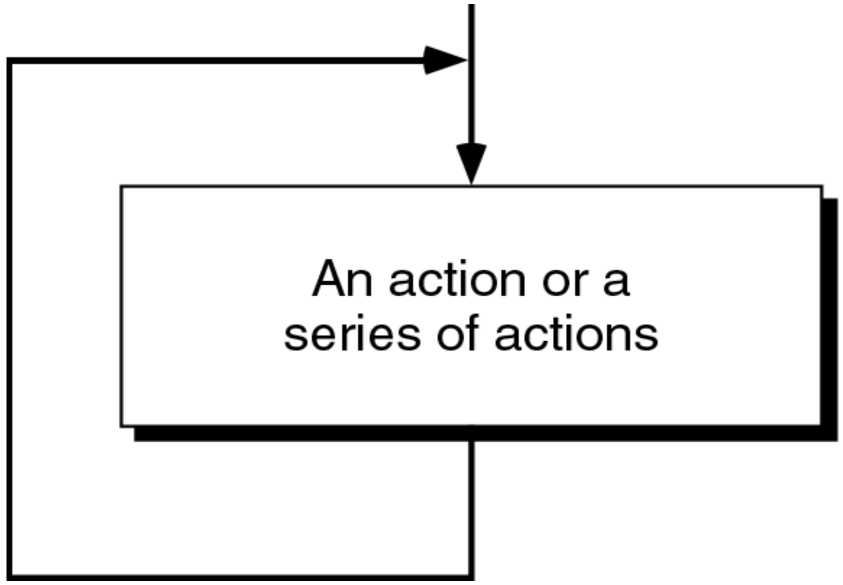
Loop

- The main idea of a loop is to repeat an action or a series of actions.



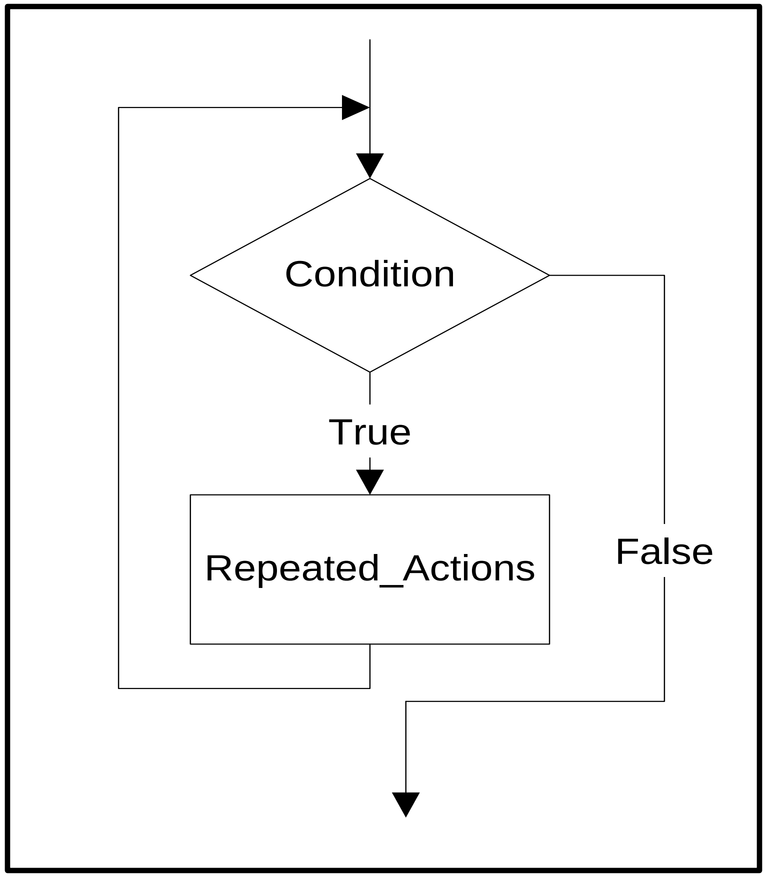
The concept of a loop without condition

- But, when to stop looping?
- In the following flowchart, the action is executed over and over again. It never stops – This is called an infinite loop
- **Solution** – put a condition to tell the loop either continue looping or stop.



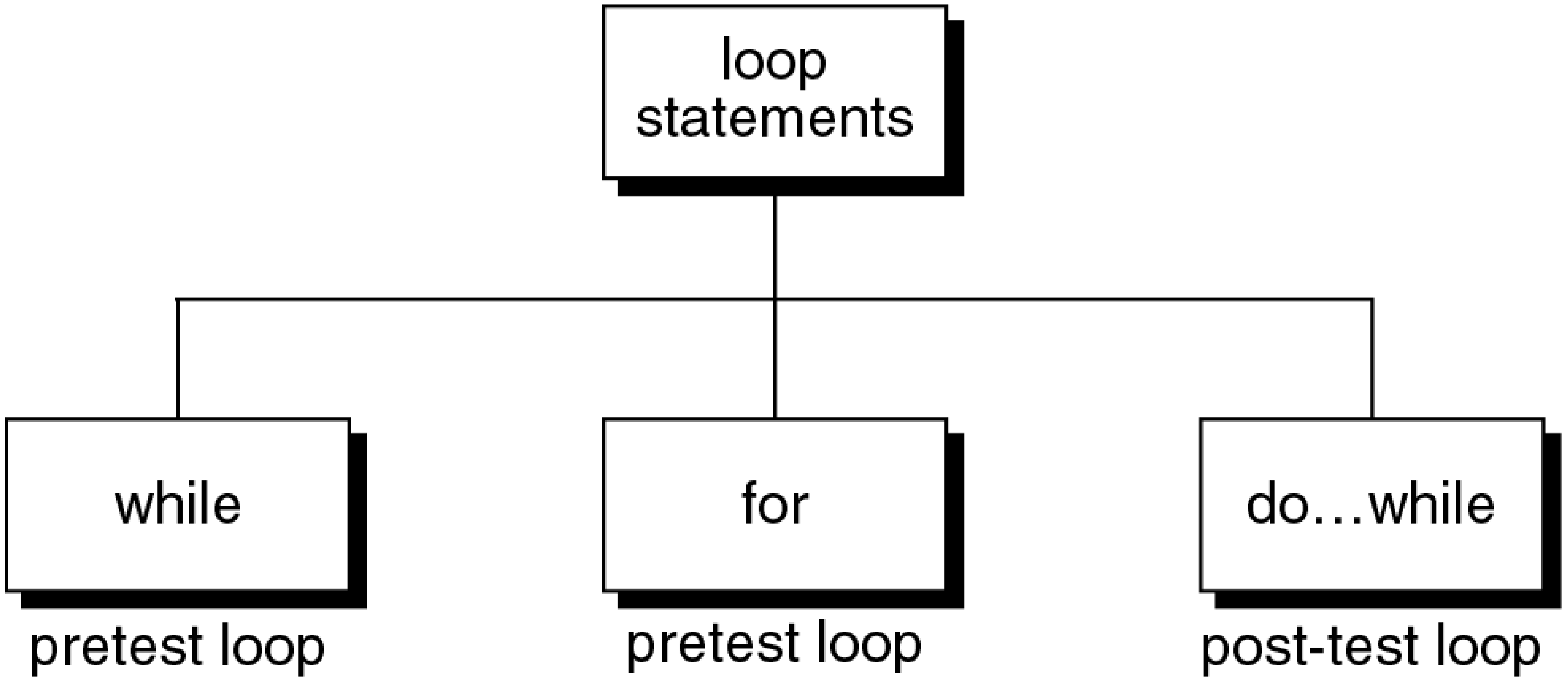
Loop

- A loop has two parts – **body** and **condition**
- **Body** – a statement or a block of statements that will be repeated.
- **Condition** – is used to control the iteration – either to continue or stop iterating.



Loop statements

- C provides three loop statements:

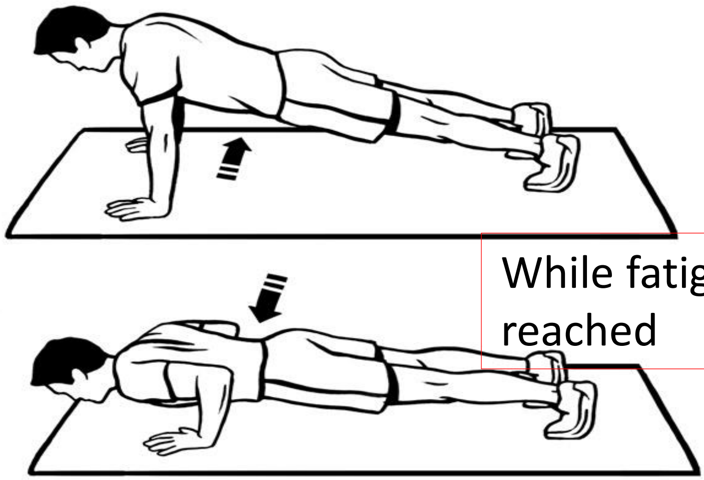


The "while" Statement in C

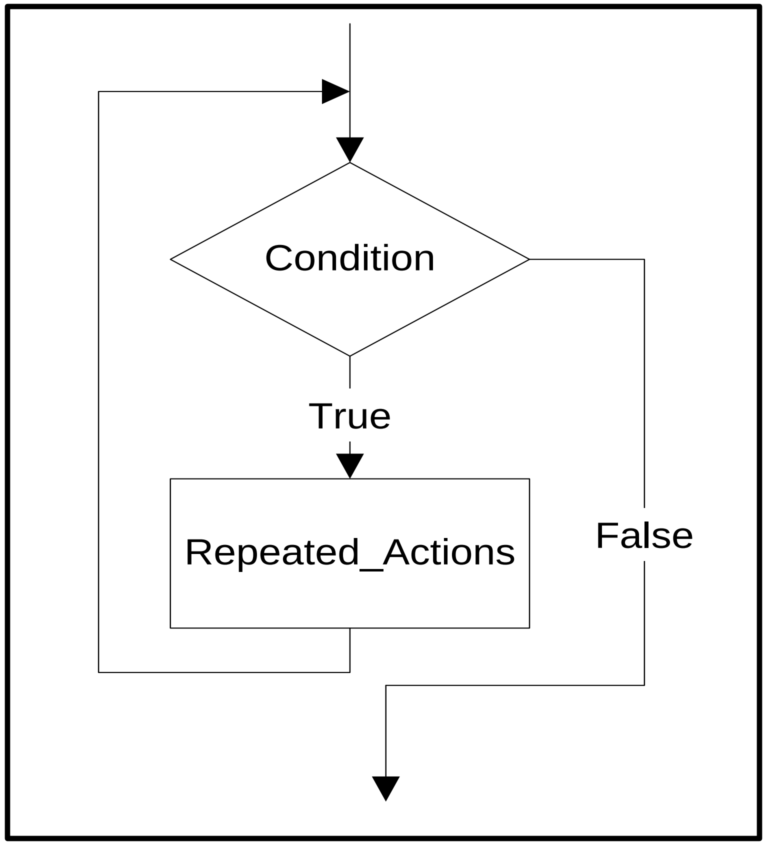
- The syntax of **while** statement in C:

Syntax

```
while (loop repetition condition)
{
   statement;
   updating control;
}
```



While fatigue level is not reached



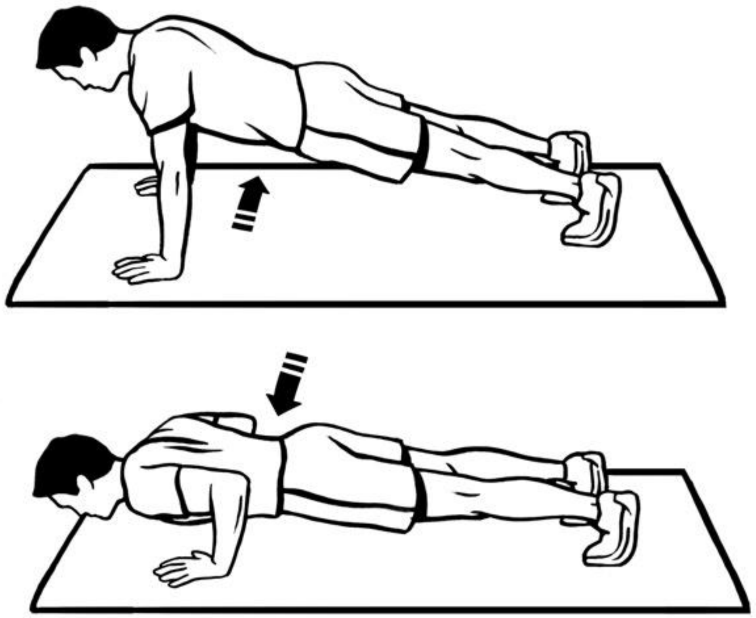
`while` statement

```
while(loop repetition condition)
{
  Statements;
}
```

Loop repetition condition is the condition which *controls the loop.*

- The ***statement*** is *repeated as long as the loop repetition condition is* ***true****.*
- A loop is called an **infinite loop** if the loop repetition condition is always true.

Example: This while statement prints numbers 10 down to 1

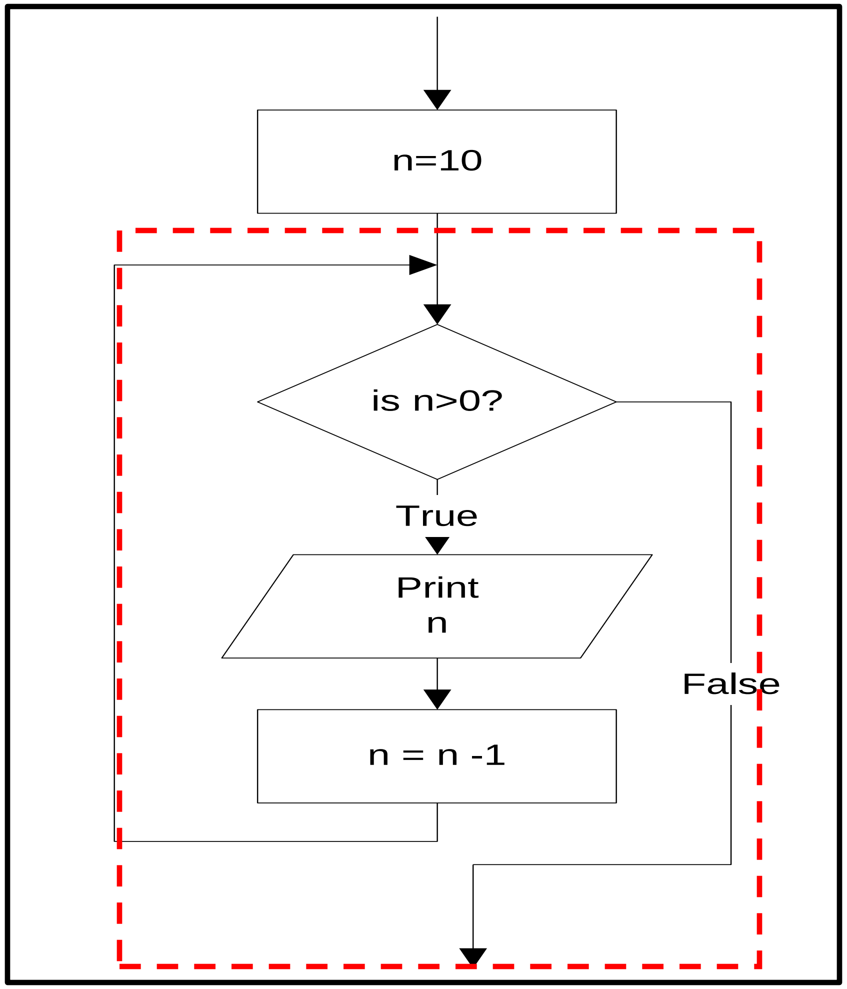


Do TEN push ups imposes a count condition

while statement

```
#include<stdio.h>
int main()
{
 int n=10;
 while (n>0){
  printf("%d ", n);
  n=n-1;
 }
}
```

```
10 9 8 7 6 5 4 3 2 1
```



The for Statement in C

- The syntax of `for` statement in C:

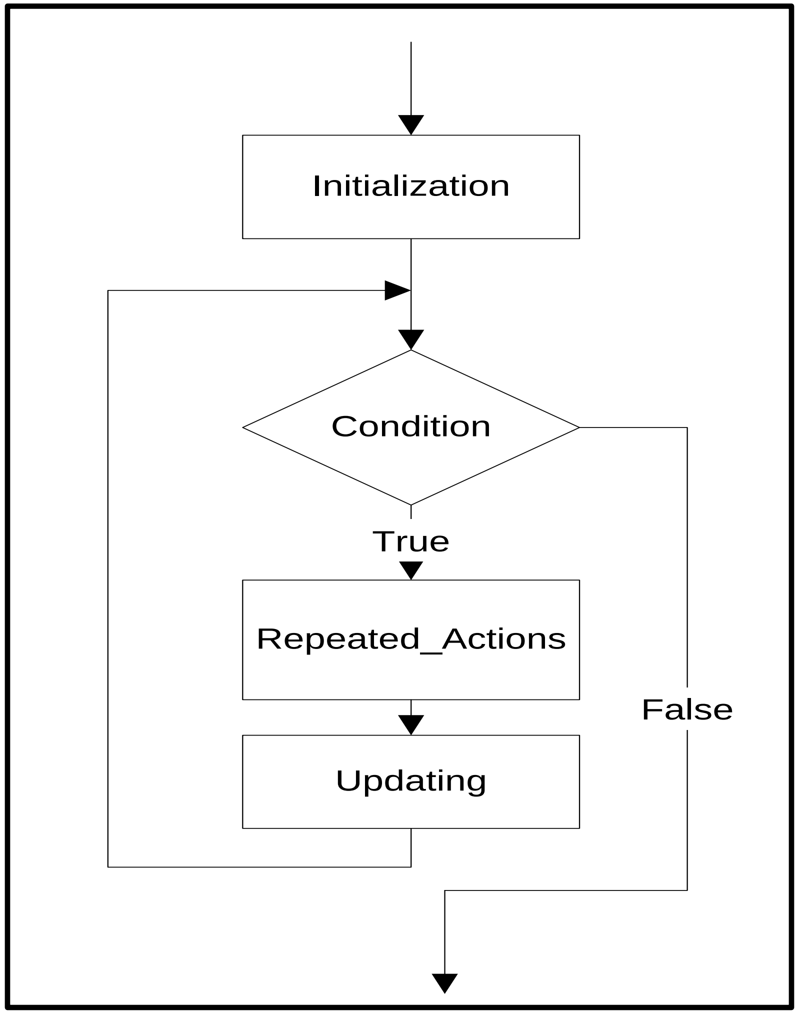
Syntax

```
for (initialization-expression;
        loop-repetition-condition;
        update-expression){
  statement;
}
```

- The **initialization-expression** set the initial value of the loop control variable.
- The **loop-repetition-condition** test the value of the loop control variable.
- The **update-expression** update the loop control variable.

`for` statement

```
for  (Initialization; Condition; Updating)
{
   Repeated_Actions;
}
```



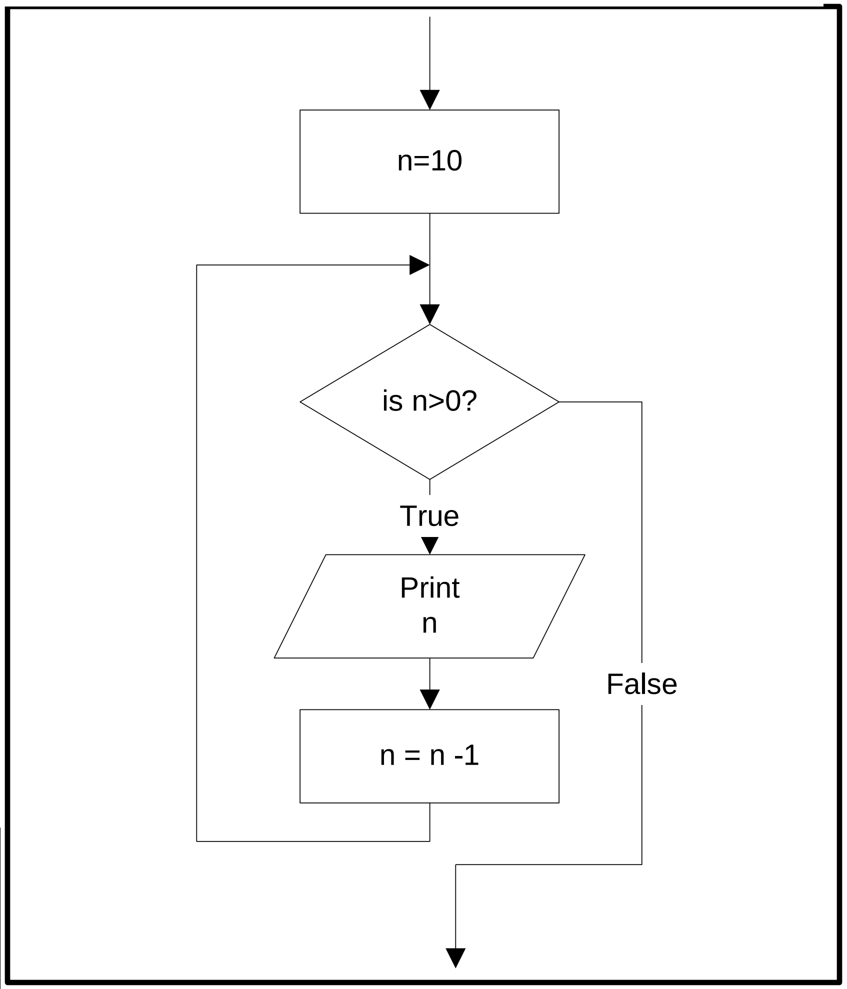
for statement

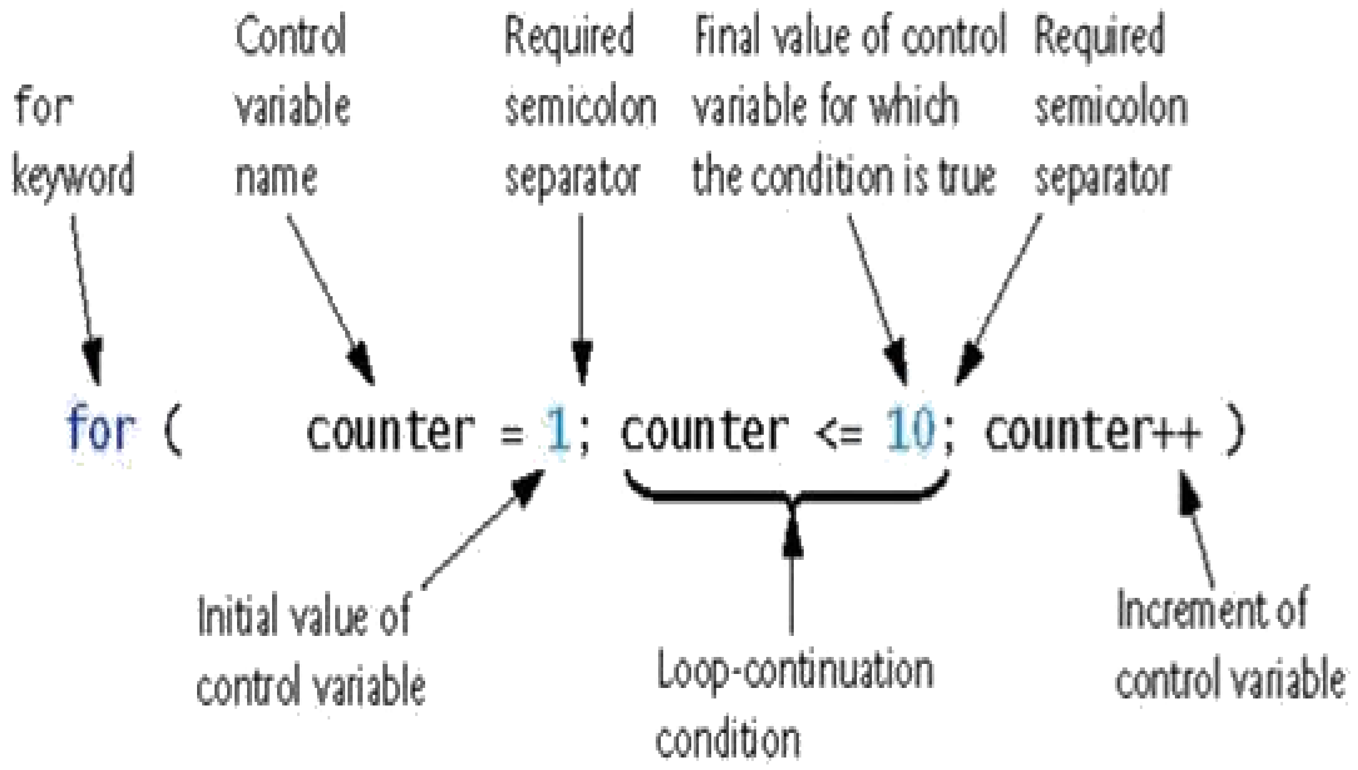
Example: This for statement prints numbers 10 down to 1

```
#include<stdio.h>
int main()
{
 int n;
 for (n=10; n>0; n=n-1){
  printf("%d ", n);
 }
}
```

```
10 9 8 7 6 5 4 3 2 1
```

Do TEN push ups = for count=1; count<=10; count++




for keyword
Control variable name
Required semicolon separator
Final value of control variable for which the condition is true
Required semicolon separator
Initial value of control variable
Loop-continuation condition
Increment of control variable

```
for (    counter = 1; counter <= 10; counter++ )
```

Nested Loops

- Nested loops consist of an **outer loop** with one or more **inner loops**.
  - Eg:

```
for (i=1;i<=100;i++){
        for(j=1;j<=50;j++){
                ...
        }
}
```

Outer loop

Inner loop

- The above loop will run for 100*50 iterations.

Program to print tables up to a given number.

```
#include<stdio.h>
void main()
{
 int i,j,k ;
 printf("Enter a number:");
 scanf("%d", &k);
 printf("the tables from 1 to %d: \n",k);
 for(i=1; i<=k; i++){
   for(j=1; j<=10; j++){
     printf("%d ",i*j);
    } //end inner for loop
   printf("\n");
 } //end outer for loop
getch();
} //end main
```

```
Enter a number
4
The tables from 1 to 4
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
```

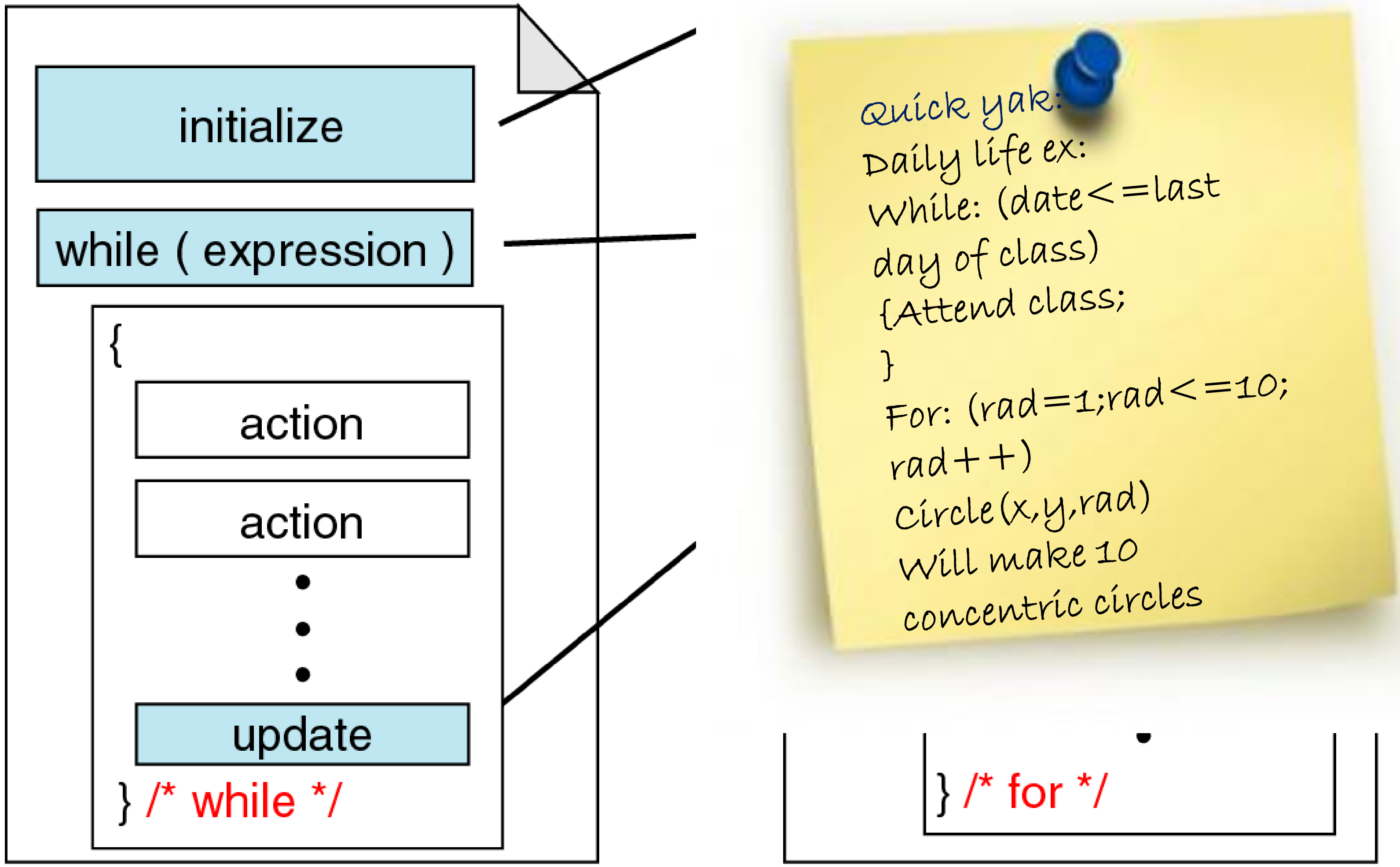
Program to display a pattern.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i,j;
 printf("Displaying right angled triangle for 5 rows");
 for(i=1 ; i<=5 ; i++)
{
      for(j=1 ; j<=i ; j++)
      {      printf("* ");}
   printf("\n");
 }
}
```

```
Displaying right angled triangle for 5 rows
*
* *
* * *
* * * *
* * * * *
```

Quick yak:
Tell them to display
various patterns, ex
reverse right angle
triangle, isometric
triangle etc.

While vs. for statements



Quick yak:
Daily life ex:
While: (date<=last
day of class)
{Attend class;
}
For: (rad=1;rad<=10;
rad++)
Circle(x,y,rad)
Will make 10
concentric circles

Comparing for and while loops

The `do-while` Statement in C

- The syntax of do-while statement in C:

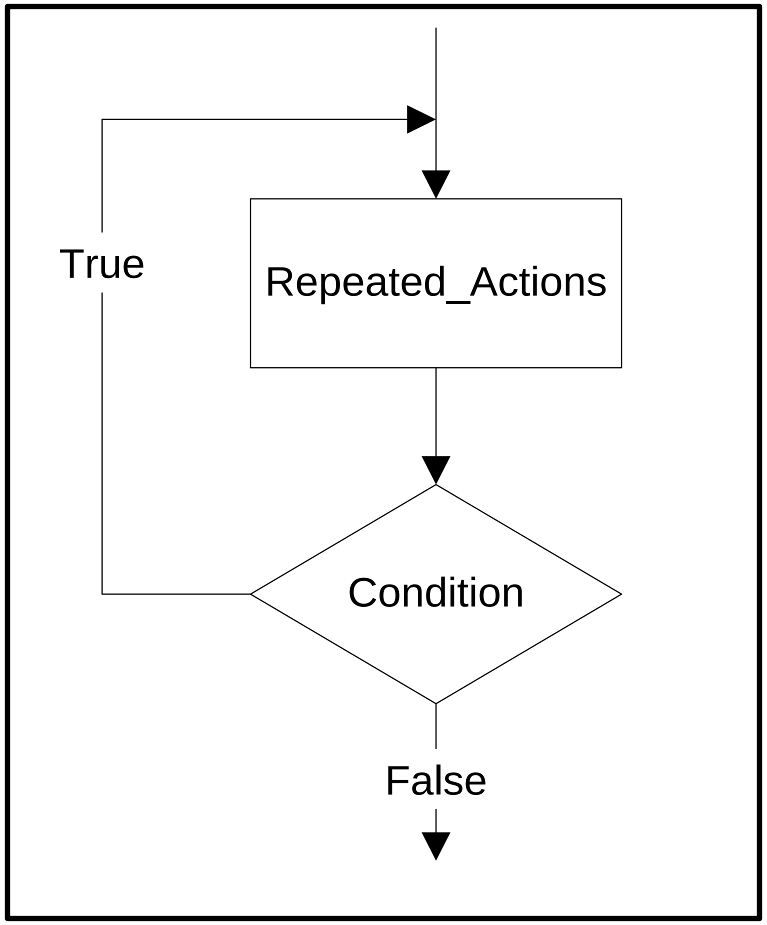
Syntax

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

- The *statement* executed at least one time.
- For second time, If the **condition** is true, then the *statement* is repeated else the loop is exited.

do…while statement

```
do
{
   Repeated_Actions;
} while (Condition);
```

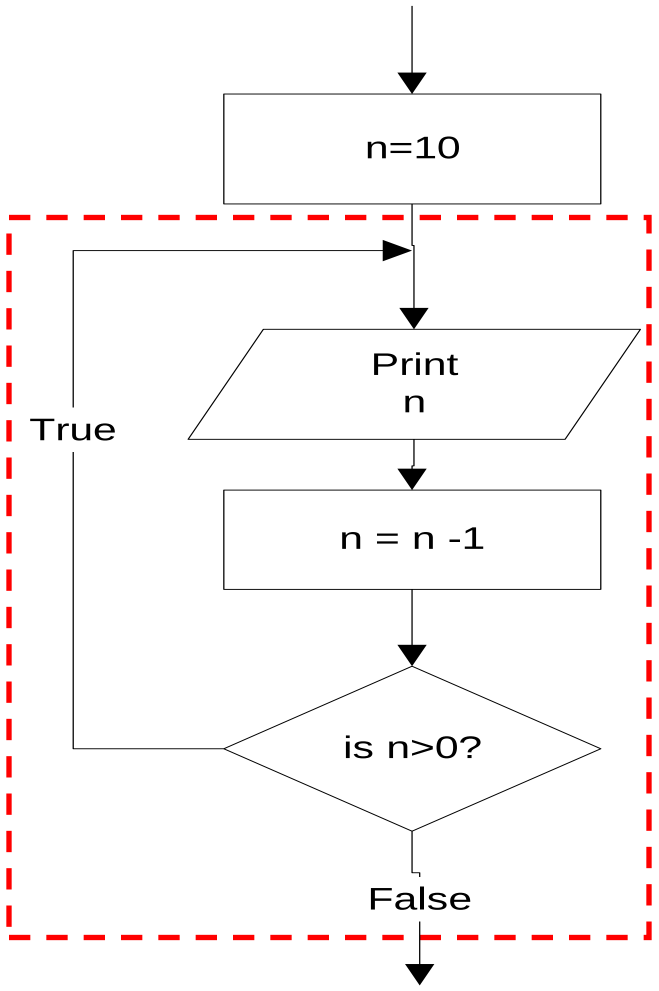


do…while statement

Example: this do…while statement prints numbers 10 down to 1

```
#include<stdio.h>
int main()
{
 int n=10;
 do
{
  printf("%d ", n);
  n=n-1;
 }
while (n>0);
}
```

```
10 9 8 7 6 5 4 3 2 1
```



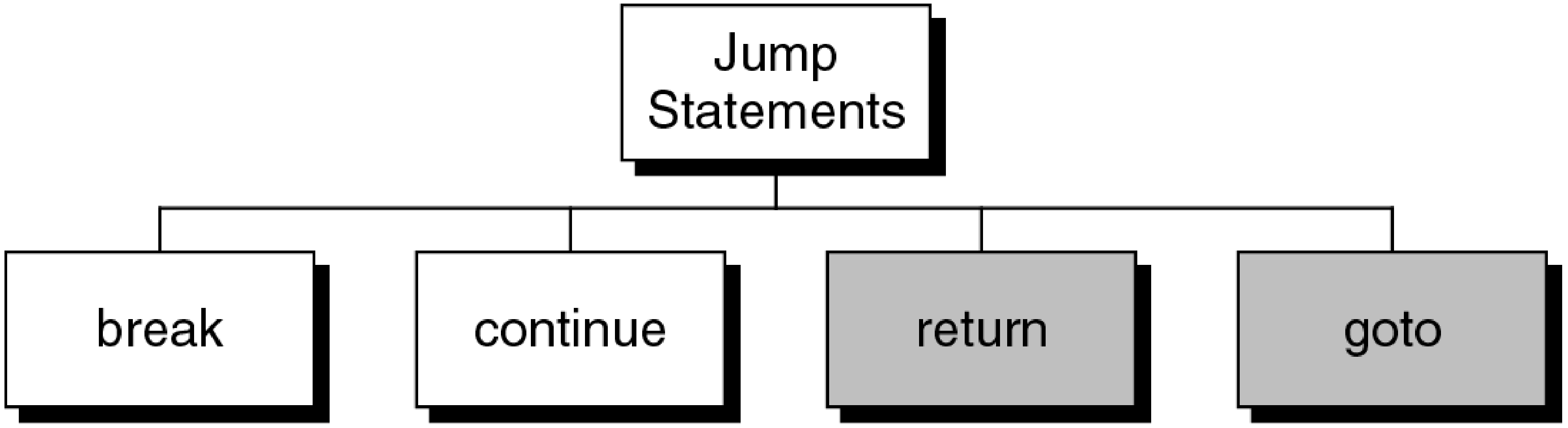
Difference between while and do..while

while loop	do..while loop
1. Condition is specified at the **top**	1. Condition is mentioned at the **bottom**
2. Body statements are executed when the condition is satisfied	2. Body statements are executed at least once even if the expression value evaluates to false
3. It is an **entry controlled** loop	3. It is an **exit controlled** loop

4.Syntax:
while (condition)
statement;
	4.Syntax:
	do
	{
	statements;
	}
	while (condition);

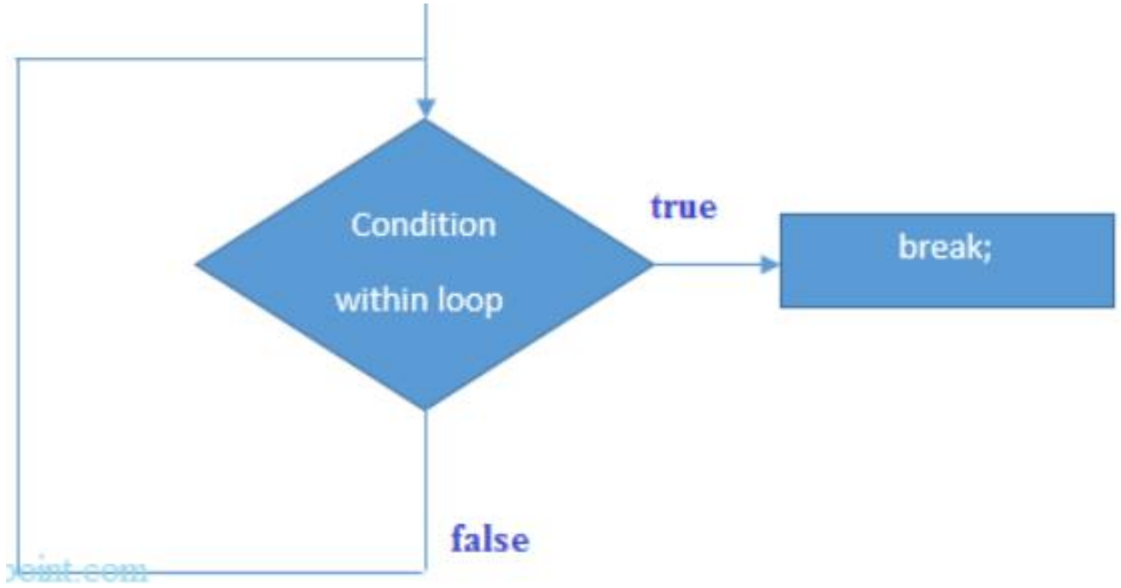
Jump statements

- You have learn that, the repetition of a loop is controlled by the loop condition.
- C provides another way to control the loop, by using **jump statements.**
- There are four jump statements:



`break` statement

- `break` is a keyword.
- `break` allows the programmer *to **terminate** the loop.*



break statement

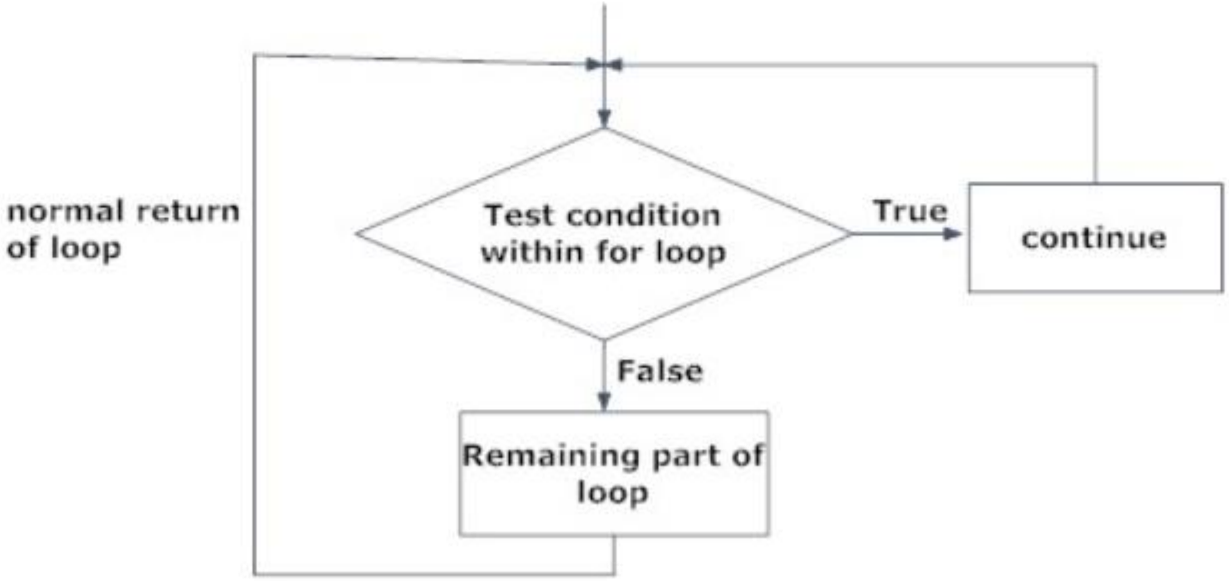
```
#include<stdio.h>
int main()
{
 int n;
for (n=10; n>0; n=n-1)
{
   if (n<8)
    break;
   printf("%d ", n);
  } //end for
}
```

```
10 9 8
```

Program to show use of break statement.

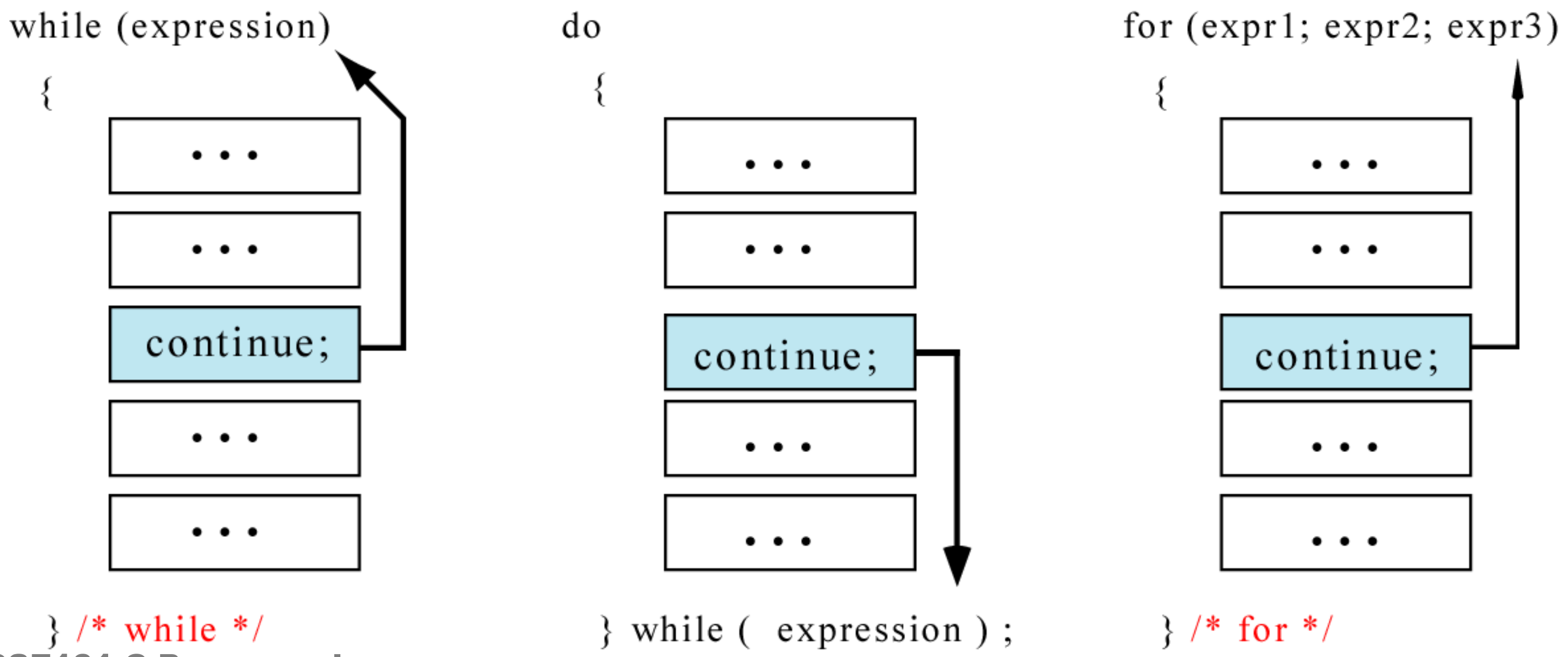
`continue` statement

- `continue` statement is ***exactly opposite to*** **`break`**.
- `continue` statement is used for *continuing the next iteration of the loop statements*
- When it occurs in the loop, it does not terminate, but skips the statements after this statement



`continue` statement

- In `while` and `do…while` loops, the continue statement transfers the control to the loop condition.
- In `for` loop, the continue statement transfers the control to the updating part.



continue statement

```
#include<stdio.h>
int main()
{
 int n;
 for (n=10; n>0; n=n-1){
  if (n%2==1)
     continue;
   printf(“%d ”, n);
 }
}
```

```
10 8 6 4 2
```

Program to show the use of continue statement in for loop

continue statement

```
#include<stdio.h>
int main()
{
 int n = 10;
 while(n>0){
  printf("%d", n);
  if (n%2==1)
    continue;
  n = n -1;
 }
}
```

For n=9, loop goes to infinite execution

Program to show the use of continue statement in for loop

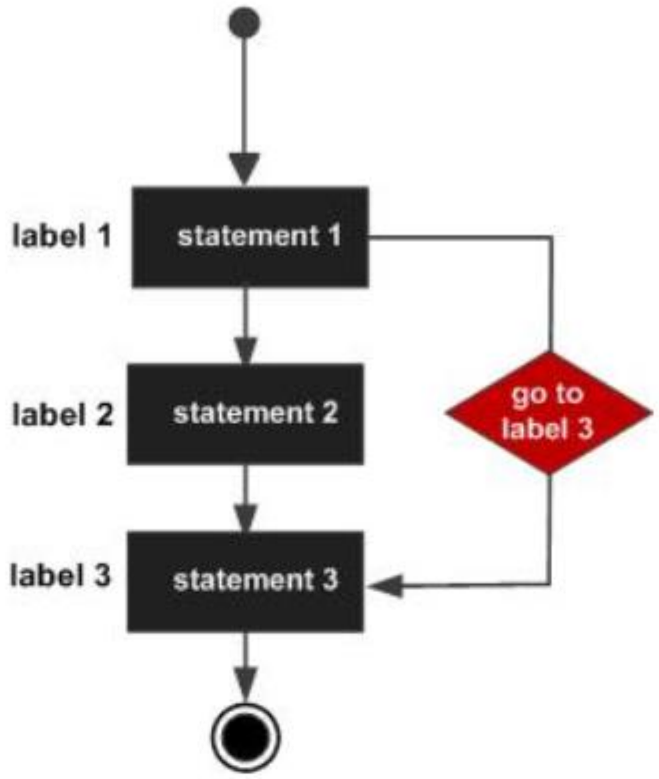
```
10 9 9 9 9 9 …………
```

The loop then prints number 9 over and over again. It never stops.

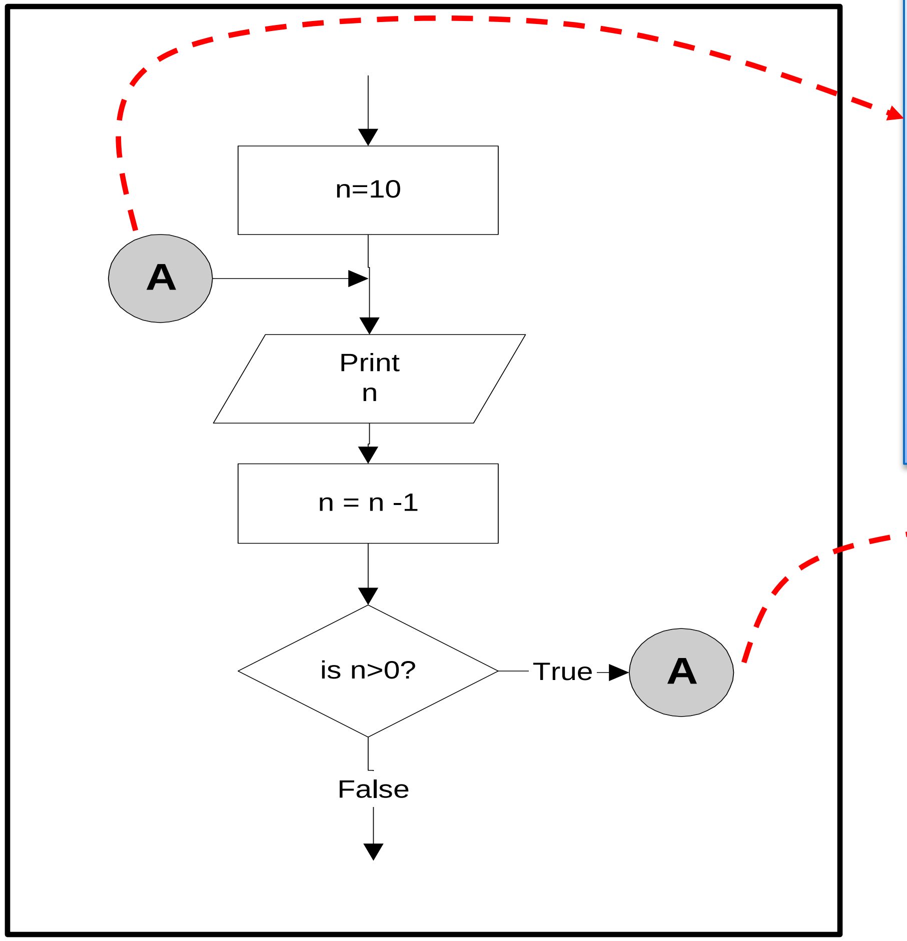
Goto statement

- **Unconditionally transfer control.**
- `goto` may be used for ***transferring control from one place to another***.
- The syntax is:

  `goto` *identifier*;



goto statement



```
n=10;

A:
   printf("%d ", n);
   n = n -1;

   if (n>0)
        goto A;
```

```
Output:

10 9 8 7 6 5 4 3 2 1
```

Program to show goto statement.

```
#include<stdio.h>
void main()
{
 int x;
 printf("enter a number: ");
 scanf("%d",&x);
 if(x%2==0)
       goto even;
 else
       goto odd;
 even:
   printf(" %d is even", x);
   return;
 odd:
   printf("%d is odd", x);
}
```

```
enter a number: 18
18 is even
```

`return` statement

- **Exits the function.**
- The return value could be any valid expression that returns a value:

i. a constant

ii. a variable

iii. a calculation, for instance (a + b) * c

iv. call to another function that returns a value

- The syntax is:

return [expression];

For example,

```
int sqr (int x){
    return (x*x);
}
```

Next Class: Formatted and Unformatted Input / Output functions

cse101@lpu.co.in