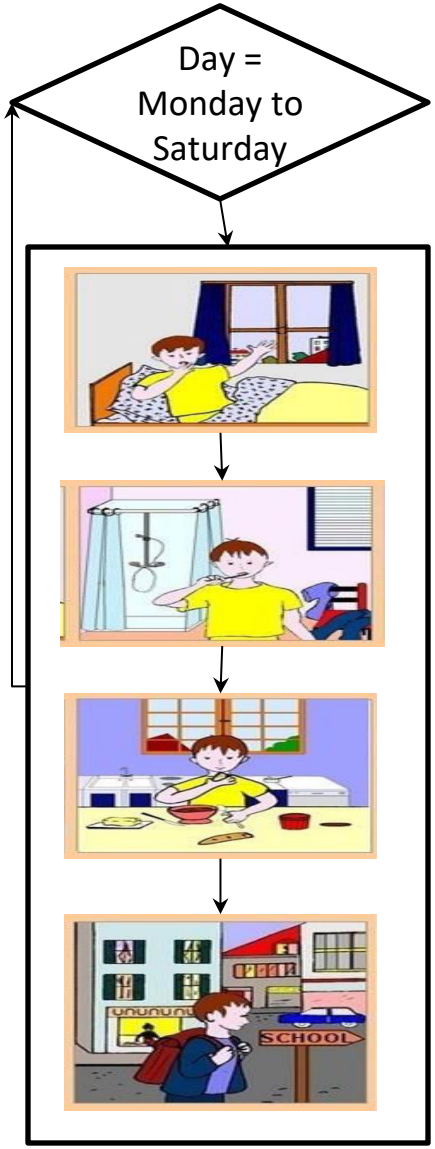
CSE101-Lec#6-Part-2

- Control Structures[Repetition structures/ or Looping statements/ or Iterative statements]

Outline

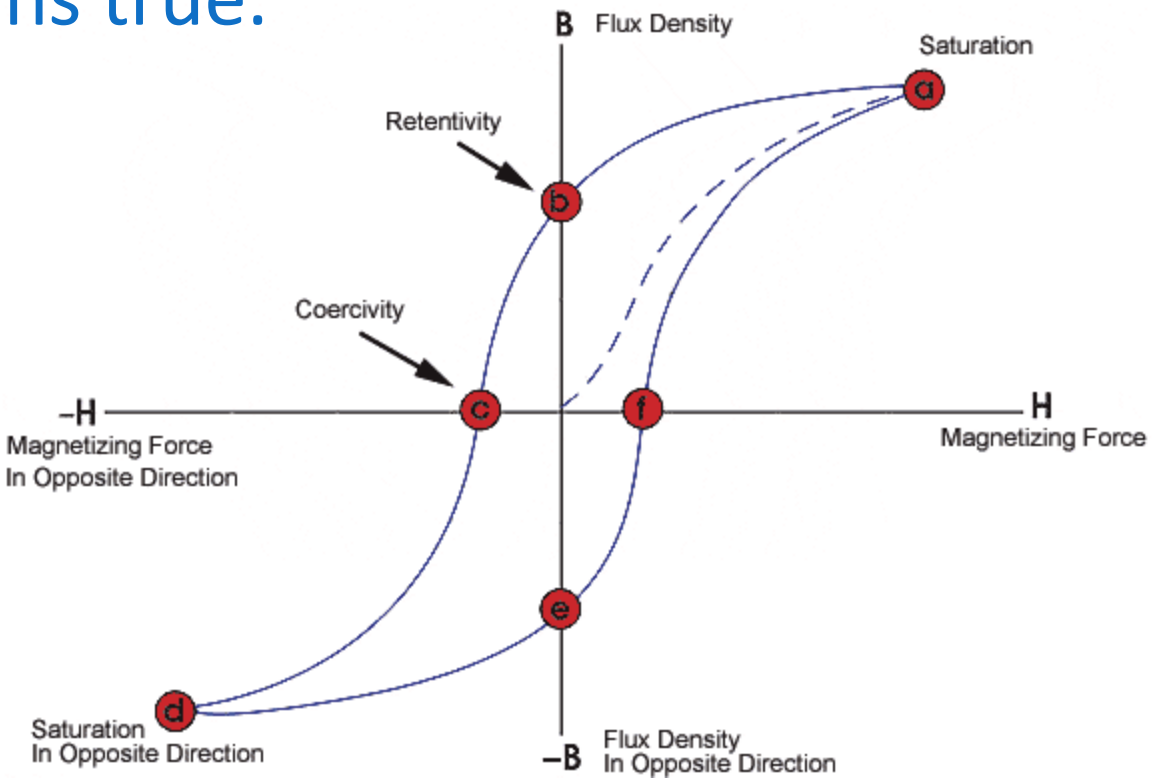
- Repetition structure/Control Loop Statements
  - for statement
  - while statement
  - do-while statement

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.



Looping (repetition)

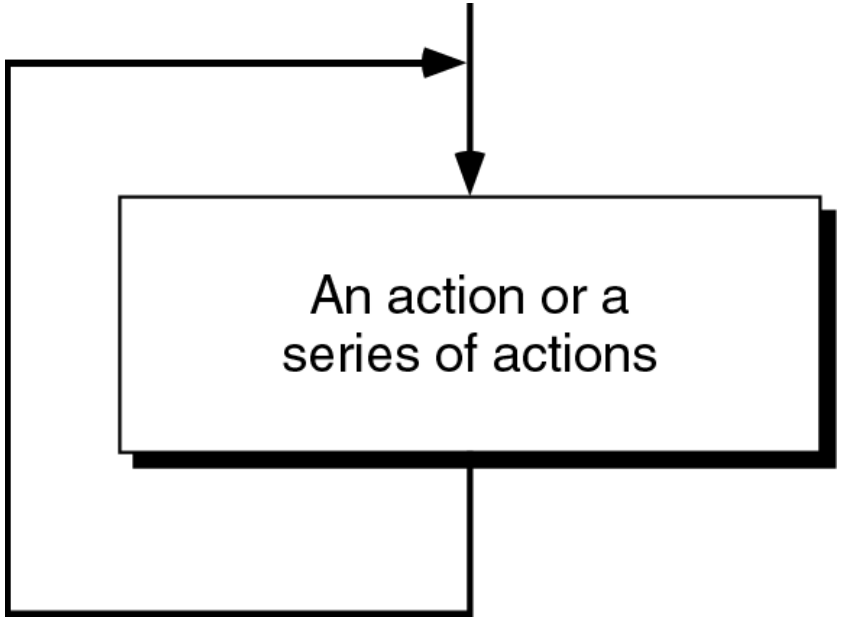
- *What if we want to display hello 500 times?*
  - Should we write 500 printf statements or equivalent ?

 Obviously not.

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

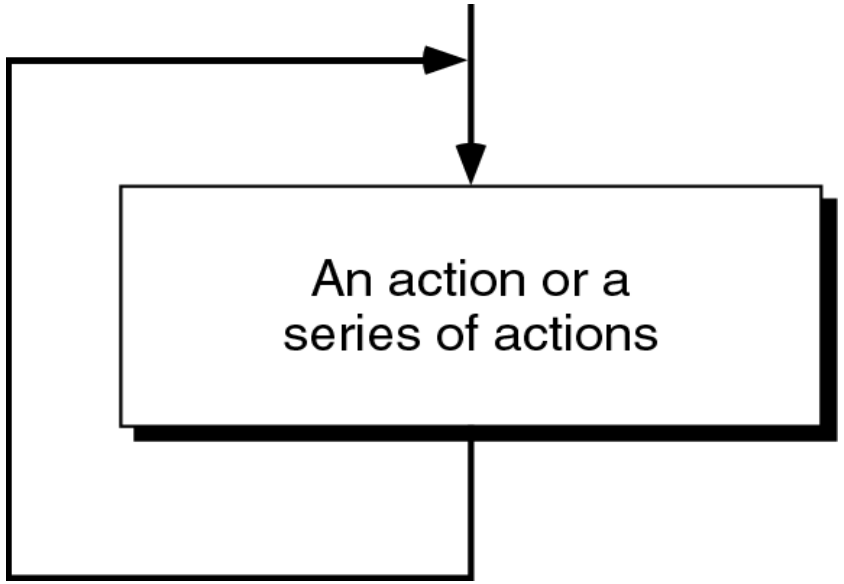
Loop

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



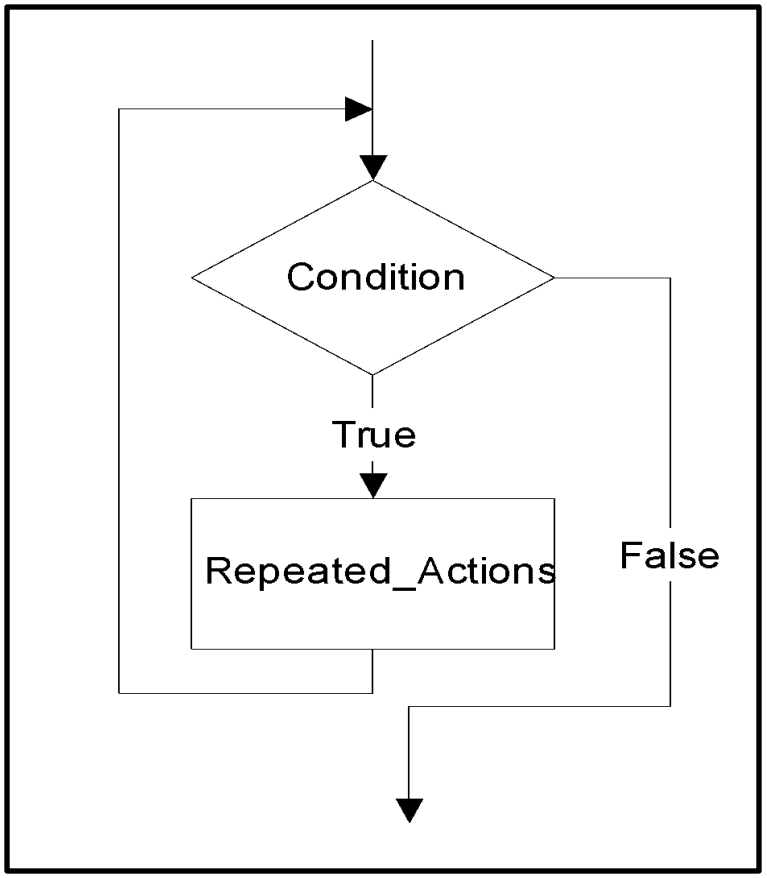
The concept of a loop without condition

- 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.



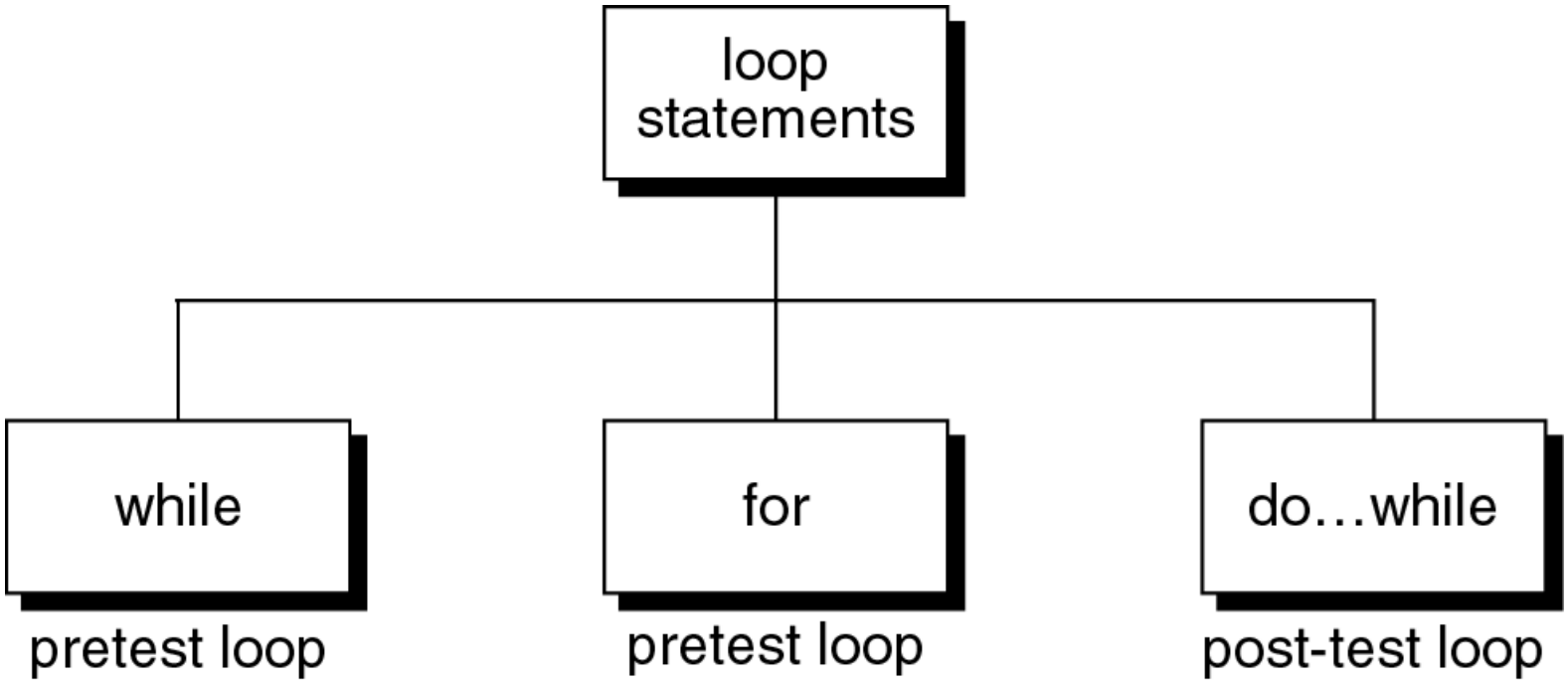
Loop

- 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:

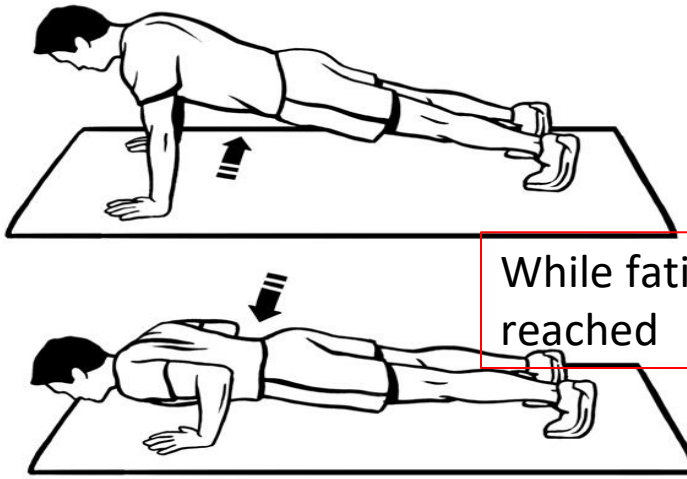


The "while" Statement in C

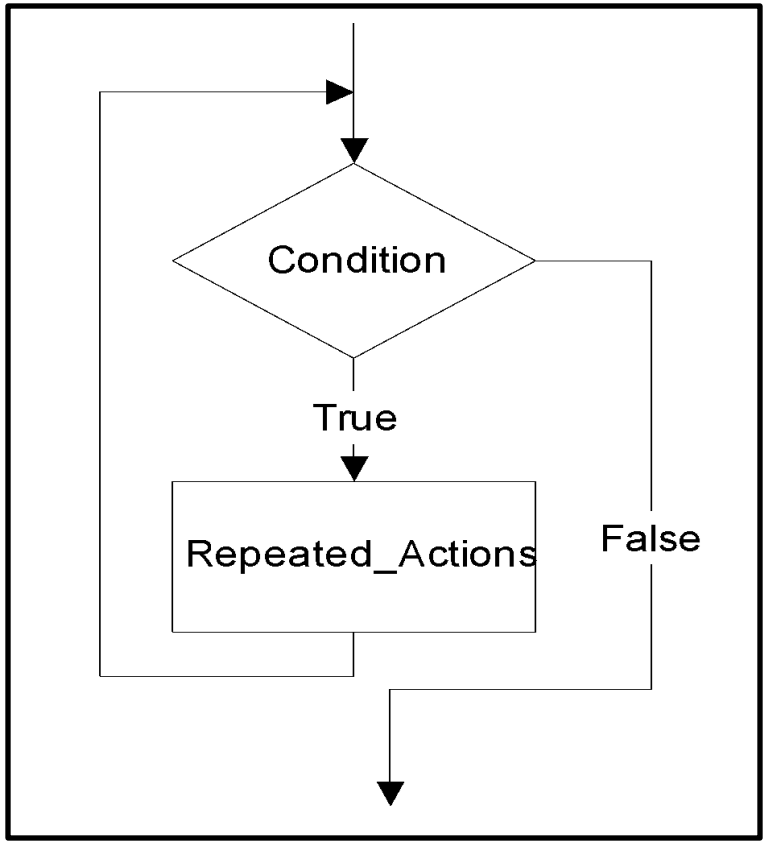
- The syntax of **while** statement in C:

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.

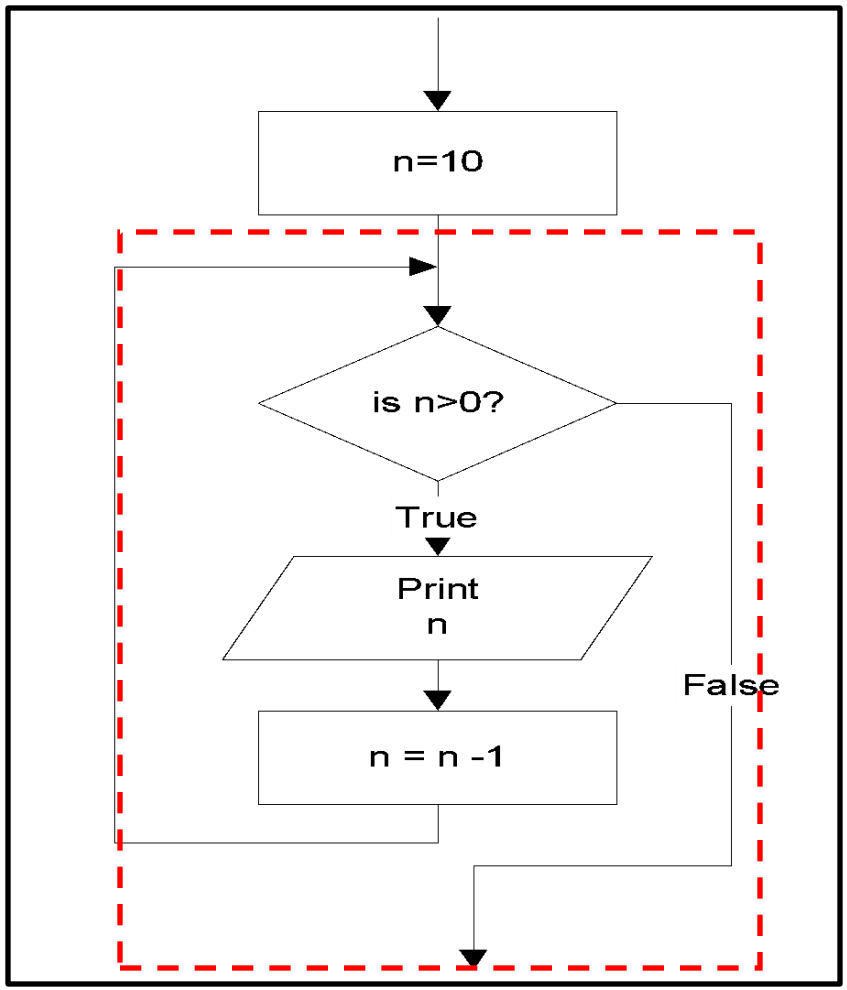
- The ***statement*** is repeated as long as the loop repetition condition is **true**.
- A loop is called an **infinite loop** if the loop repetition condition is always true.
- while loop is known as entry controlled loop, as condition is checked at the beginning/ or entry point

while statement

Example: This while statement prints numbers 10 down to 1

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

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



count condition

Q1

How many times i value is checked in the following C code?

```
#include <stdio.h>
int main()
{
    int i = 0;
    while (i < 3)
        i++;
    printf("In while loop\n");
}
```

A. 2

B. 3

C. 4

D. 1

Q2

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 0;
    while (++i)
    {
        printf("H");
    }
    return 0;
}
```

A. H

B. H is printed infinite times

C. Compile time error

D. Nothing will be printed

Q3

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 0;
    while (i = 0)
        printf("True\n");
    printf("False\n");
    return 0;
}
```

A. True (infinite time)

B. True (1 time) False

C. False

D. Compiler dependent

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 0, j = 0;
    while (i < 5, j < 10)
    {
        i++;
        j++;
    }
    printf("%d, %d\n", i, j);
    return 0;
}
```

A. 5, 5

B. 5, 10

C. 10, 10

D. Compiler error

Q5

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i=0;
    while(++i<=5);
    printf("%d ",i);
    return 0;
}
```

A. 1 2 3 4 5

B. 6

C. 5

D. Compiler error

The for Statement in C

- The syntax of `for` statement in C:

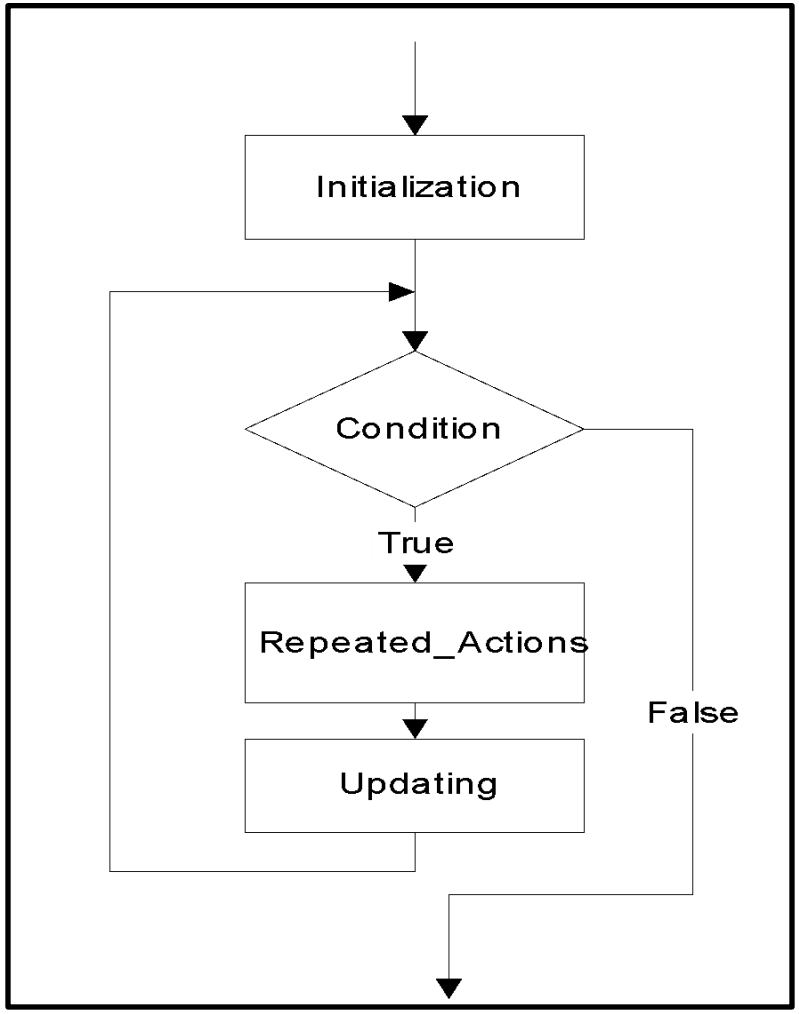
Syntax

```
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.
- It is also known as entry controlled loop as condition is checked first and then loop body executes

for statement

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

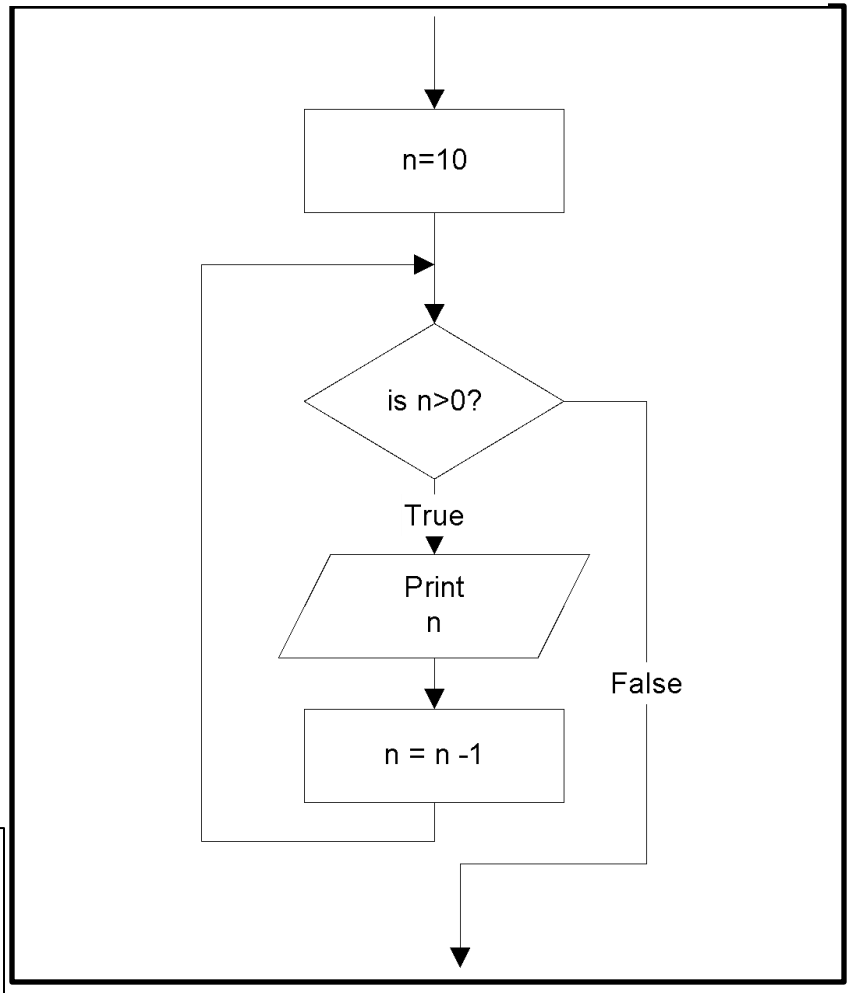


for statement

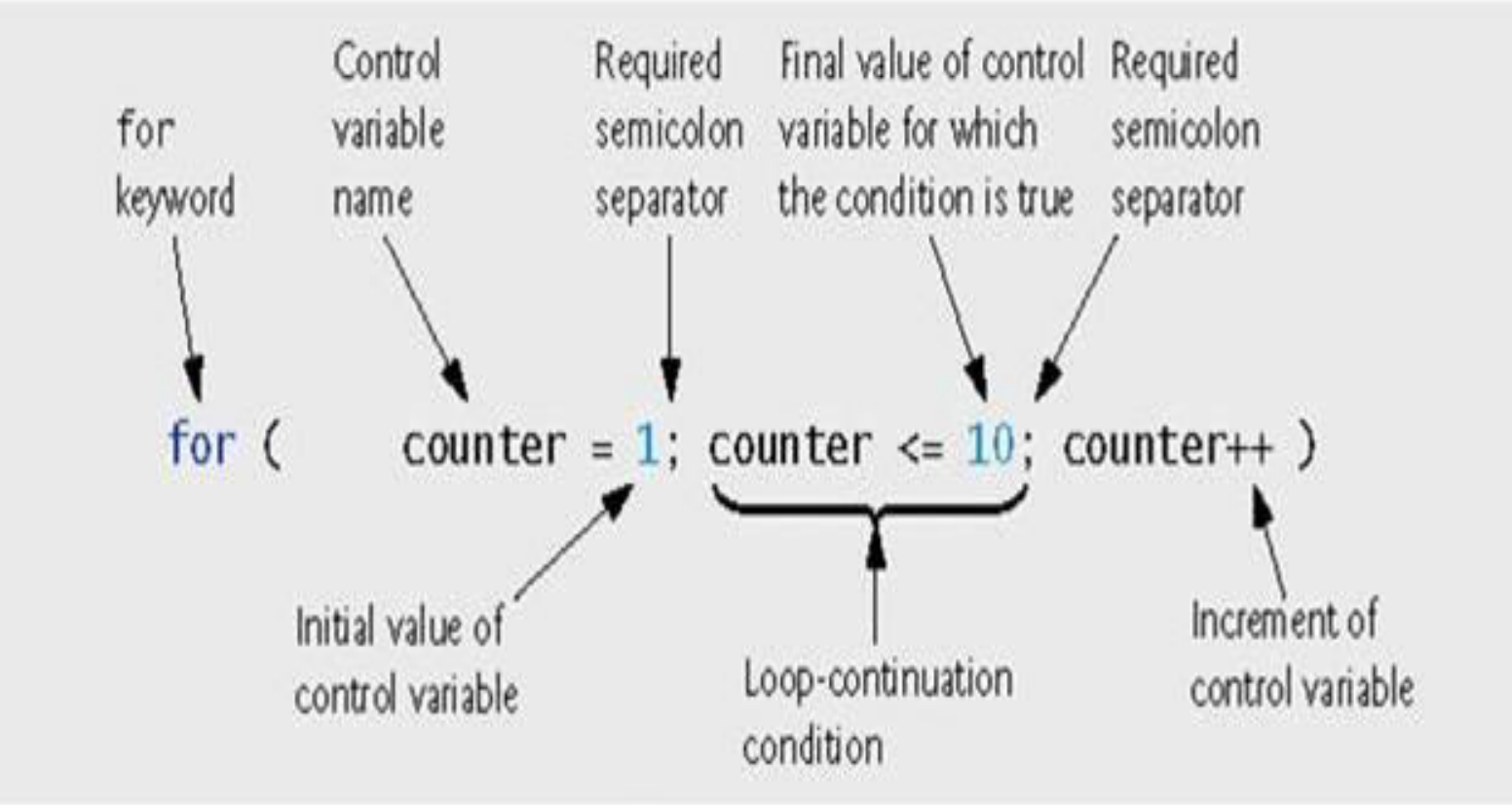
Example: This for statement prints numbers 10 down to 1

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

```
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
for (counter = 1; counter <= 10; counter++)
Initial value of control variable
Loop-continuation condition
Increment of control variable

Q1

```
#include <stdio.h>
int main()
{
  int i;
  for (i = 1; i != 10; i += 2)
    printf("Hello");
  return 0;
}
```

A. Hello will be displayed 5 times

B. Hello will be displayed 4 times

C. Hello will be displayed infinite no. of times.

D. Hello will be displayed 6 times

Q2

What will be the output of following code

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

A. Numbers from 1 to 9 will be printed
B. 10
C. 9
D. Infinite loop

Q3

What will be the output of following code?

```
#include<stdio.h>
int main()
{
int i;
for(i=2;i<=10;)
{
   printf("%d ",++i);
}
return 0;
}
```

A. 2 3 4 5 6 7 8 9 10
B. 3 4 5 6 7 8 9 10 11
C. infinite loop
D. Compile time error

Q4

What will be the output of following code?

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

A. 1
B. 0
C. infinite loop
D. Nothing will be displayed

Q5

What will be the output of following code?

```
#include<stdio.h>
int main()
{
int i,j;
for(i=1,j=1;j<=5;j++)
{
}
printf("\n%d %d",i,j);
return 0;
}
```

A. 1 6

B. 1 1

C. 6 1

D. 1 5

for vs while loop

FOR LOOP	WHILE LOOP
Initialization may be either in loop statement or outside the loop.	Initialization is always outside the loop.
Once the statement(s) is executed then after increment is done.	Increment can be done before or after the execution of the statement(s).
It is normally used when the number of iterations is known.	It is normally used when the number of iterations is unknown.
Condition is a relational expression.	Condition may be expression or non-zero value.
It is used when initialization and increment is simple.	It is used for complex initialization.
For is entry controlled loop.	While is also entry controlled loop.
for (init ; condition ; iteration) { statement(s); }	while (condition) { statement(s); }

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>
int 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
return 0;
} //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
```

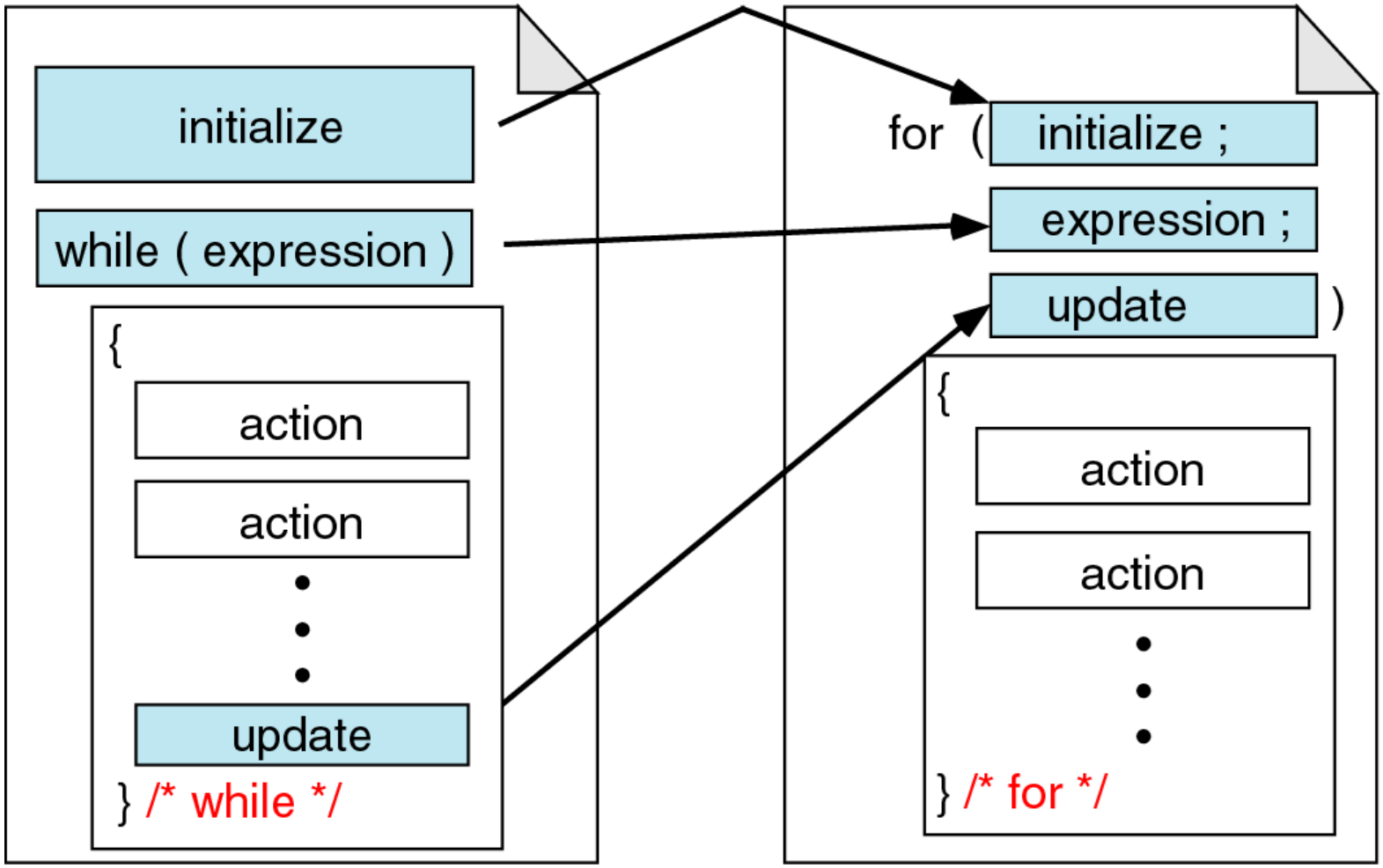
Program to display a pattern.

```
#include<stdio.h>
int 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");
 }
return 0;
}
```

Displaying right angled triangle for 5 rows

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

While vs. for statements



Comparing `for` and `while` loops

The `do-while` Statement in C

- The syntax of do-while statement in C:

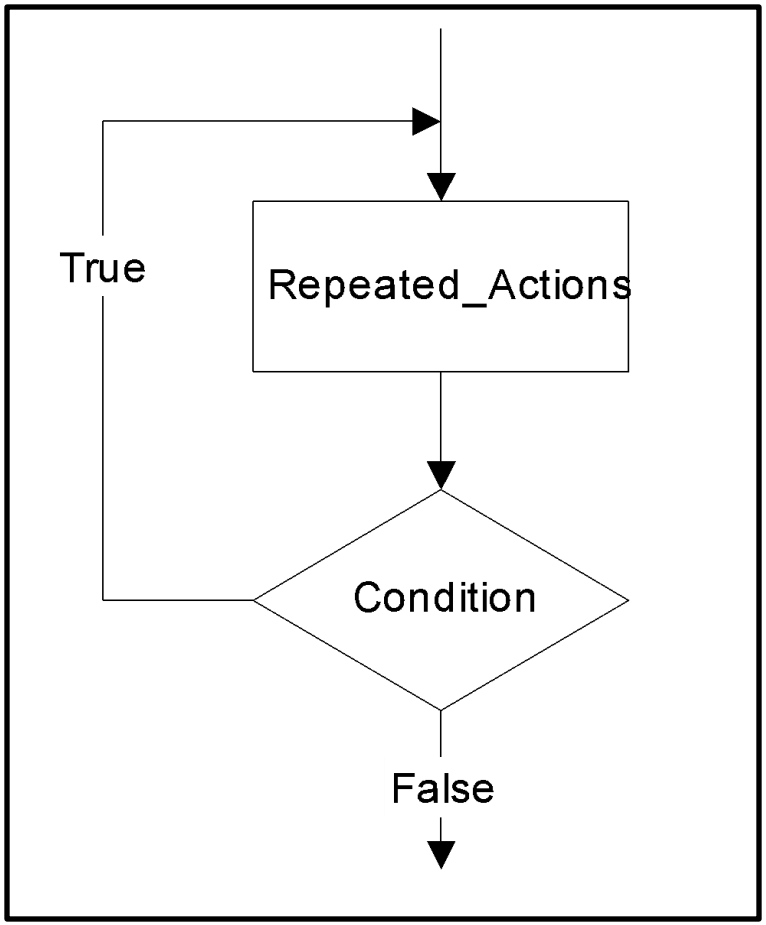
Syntax

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

- The *statement* executed at least one time(even if the condition is false)
- For second time, If the **condition** is true, then the *statement* is repeated else the loop is exited.
- Also known as exit-controlled loop, as loop body executes first and then the condition is checked

do...while statement

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

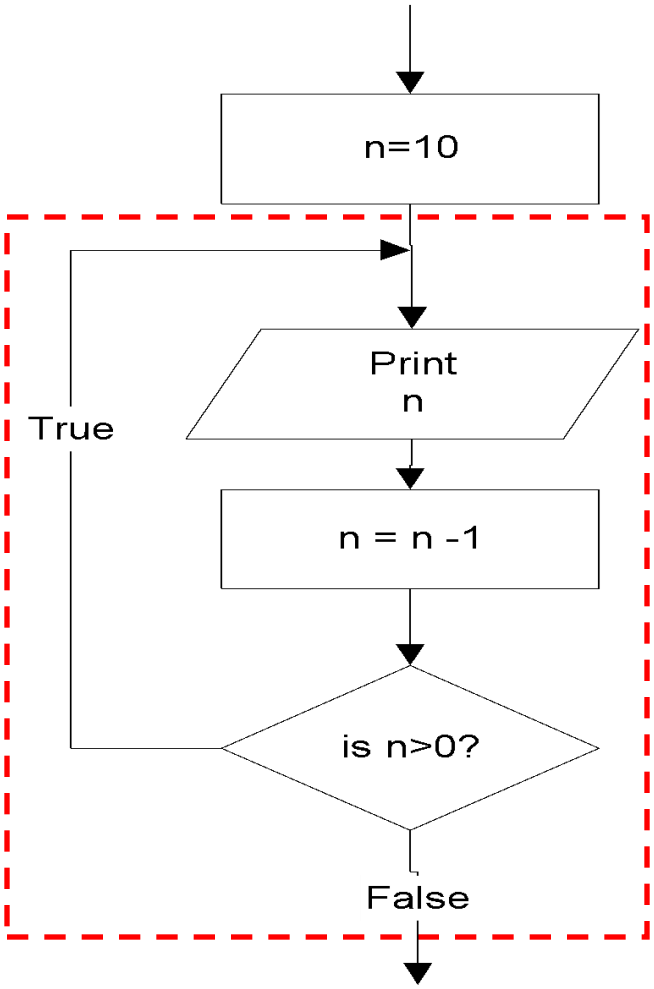


do…while statement

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);**

Q1

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    do
        printf("In while loop ");
    while (0);
        printf("\nAfter loop");
    return 0;
}
```

A. In while loop

B. In while loop
After loop

C. After loop

D. Infinite loop

Q2

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 0;
    do {
        i++;
        printf("In while loop\n");
    } while (i < 3);
   return 0;
}
```

A. In while loop
In while loop
In while loop

B. In while loop
In while loop

C. Nothing will be displayed

D. Compile time error

Q3

How many times i value is checked in the following C code?

```
#include <stdio.h>
int main()
{
   int i = 0;
   do {
      i++;
      printf("in while loop\n");
   } while (i < 3);
   return 0;
}
```

A. 2

B. 3

C. 4

D. 1

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 0;
    do
    {
        printf("Hello");
    } while (i != 0);
    return 0;
}
```

A. Nothing

B. H is printed infinite times

C. Hello

D. Run time error