C Programming:

Course 6:

ITERATIVE Statements

Dr. Chénche

Introduction :

Exercise

Write a program that displays the number 1 ten times.

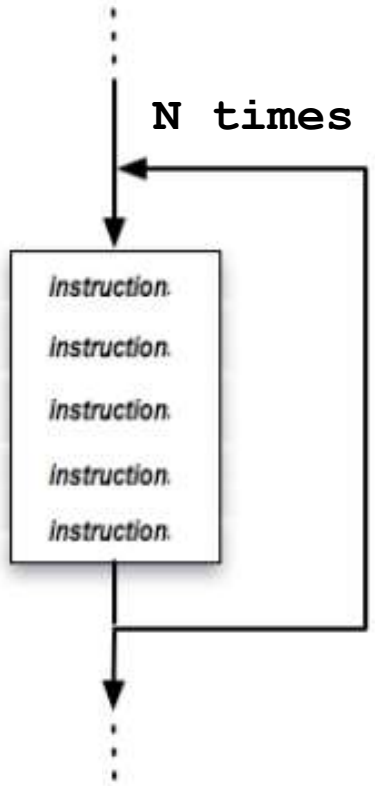
Write a program that displays numbers from 1 to 100.

In computer programming, loops are used to repeat a block of code.

For example, if we want to show a message 100 times, then instead of writing the printf statement 100 times, we can use a loop.

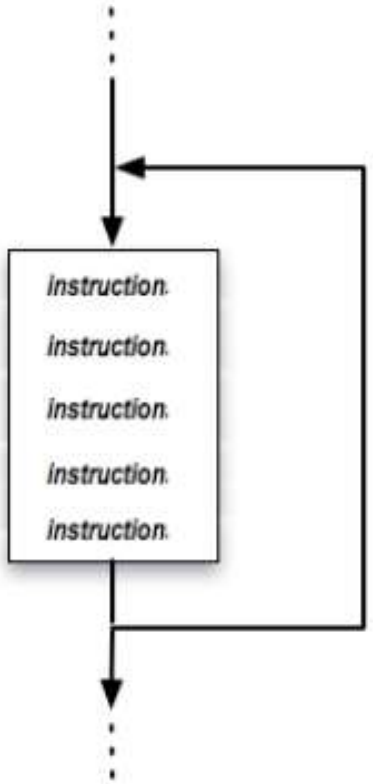
What is a Loop?

A loop is a control structure that allows you to repeat the same instructions multiple times.



What is a Loop?

A loop is a control structure that allows you to repeat the same instructions multiple times.



There are three common loop types in C:

1. **while**
2. **do... while**
3. **for**

An iteration is the term used for each pass through a loop.

`for` Loop

Syntax:

```
for( initialization ; condition ; modification )
...
block of statements;
...
```

- The 'for' loop is a compact loop structure that combines loop initialization, condition testing, and incrementing the loop variable.
- It's very suitable when you know in advance how many times you want to iterate.

Example of a "for" Loop

Here's an example using a 'for' loop to print the variable 'i' from 0 to 9.

Example:

```
for (int i = 0; i < 10; i++)
{
    printf("The variable i is %d\n", i);
}
```

Example of Repetition

i=1

```
for  ( int i = 1 ;   i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

Example of Repetition

i=1

1<=3 yes

```
for  ( int i = 1 ;  i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

Example of Repetition

i=1

1<=3 yes

```
for  ( int i = 1 ;   i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
```

Example of Repetition

i=1

1<=3 yes

i=2

```
for  ( int i = 1 ;   i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

```
for  ( int i = 1 ;  i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

```
for  ( int i = 1 ;   i <= 3  ;  i++ )
    printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
Hello 2
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

i=3

```
for  ( int i = 1 ;   i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
Hello 2
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

i=3

3<=3 yes

```
for  ( int i = 1 ;  i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
Hello 2
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

i=3

3<=3 yes

```
for  ( int i = 1 ;   i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
Hello 2
Hello 3
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

i=3

3<=3 yes

i=4

```
for  ( int i = 1 ;   i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
Hello 2
Hello 3
```

Example of Repetition

i=1

1<=3 yes

i=2

2<=3 yes

i=3

3<=3 yes

i=4

4<=3 No

false

```
for  ( int i = 1 ;  i <= 3  ;  i++ )

     printf("Hello %d \n",i);
```

OUTPUT

```
Hello 1
Hello 2
Hello 3
```

for Loop

When the loop control condition is evaluated and has value false, the loop is said to be **"satisfied"** and control passes to the statement following the for statement.

What will be the output of this code ?

```
for (int i=0;i <= 3;i++)
printf("%d \n",i);
printf("%d",i);
```

Error

```
int i
for (i=0;i <= 3;i++)
{printf("%d \n",i);}
printf("%d",i);
```

0
1
2
3
4

`for` Loop

- **Notes:**

- A variable declared in the initialization statement (inside for loop) is visible only inside the scope of for loop.
- Don’t write semicolon after the end of parenthesis.

for Loop

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

This program displays the numbers from zero to 9

for Loop

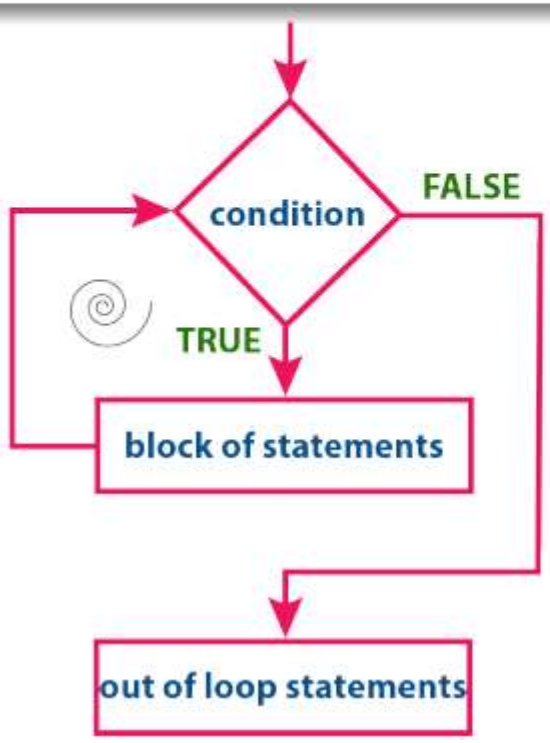
```
#include <stdio.h>
int main()
{
    int i;
    for (i = 0; i <= 100; i = i + 5)
        printf("%d\t", i);
    return 0;
}
```

What does this program ?

`while` Loop

Syntaxe :

```
while (Condition)
{
  // Block of instructions
}
```



The condition (loop control condition) is evaluated before each **iteration** of the loop.

Example of `while` Loop

Repeating a Specific Number of Times

To achieve this, we create a counter variable that starts at 0 and increments with each iteration.

Example:

```c
int counter = 0;

while (counter < 10)
{
    printf("The variable counter is %d\n", counter);
    counter++; // equivalent to counter = counter + 1;
}
```

Incrementing a variable means adding 1 to it (e.g., 'variable++;').

Beware of Infinite Loops

- **Infinite loops** happen when the loop's **condition never becomes false**.
- To avoid this, **something inside the loop must change the condition**, so that the loop can **stop after a certain number of iterations**.

Example:

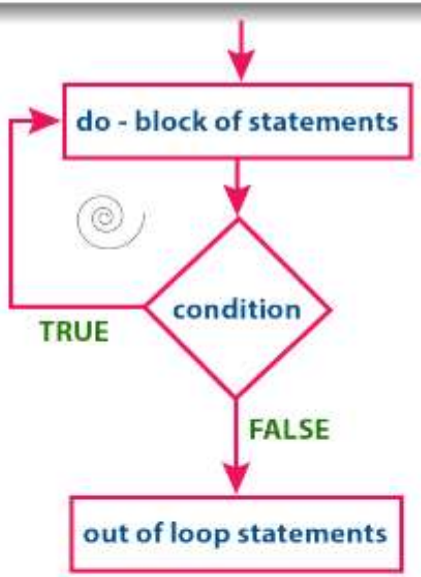
```
int counter = 0;

while (counter >= 0)
{
    printf("The variable counter is %d\n", counter);
    counter++; // equivalent to counter = counter + 1;
}
```

do..while Loop

Syntax:

```
do
{
    // Block of instructions
} while (Condition);
```



- The 'do...while' loop is very similar to 'while'.
- The main difference is the position of the condition. In 'do...while', the condition is at the end, so the loop always executes at least once.

Example `of do..while` Loop

Here's an example using a 'do...while' loop to print the variable 'n' from 0 to 9.

Example:

```
int n = 0;

do
{
    printf("The variable n is %d\n", n);
    n++;
} while (n < 10);
```

Summary of Loop Types

- The ‘while‘ loop repeats a block of instructions as long as a condition is true.
- The ‘do...while‘ loop is similar to ‘while‘, but it guarantees at least one execution of the loop body.
- The ‘for‘ loop is a compact loop structure that combines initialization, condition, and incrementing.

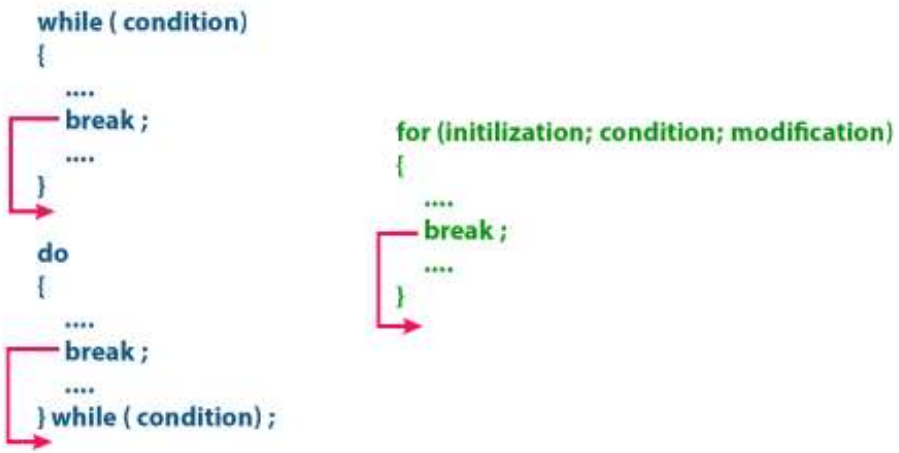
break Statement

- **break statement can be used with Switch or any of the 3 looping structures**
- **it causes an immediate exit from the Switch, while, do-while, or for statement in which it appears.**
- **if the break is inside nested structures, control exits only the innermost structure containing it**

break Statement

Example:

```
for (int i = 0; i < 10; i++)
{
    if (i == 5)
    {
        break; // exit the loop when i equals 5
    }
    printf("The variable i is %d\n", i);
}
```



break Statement

```
int j = 40;
   while (j < 80)
   {
       j += 10;
       if (j == 70)
           break;
       printf("j is %d\n", j);
   }
   printf("We are out of the loop as j = 70.\n");
```

```
j is 50
j is 60
We are out of the loop as j=70.
```

Nested loops

The block of instructions in one loop can itself contain another loop. These are called nested loops.

Example:

```
int i;
int j=1;
for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d and j=%d!\n", i, j);
    j++;
   }
}
```