

# Loops

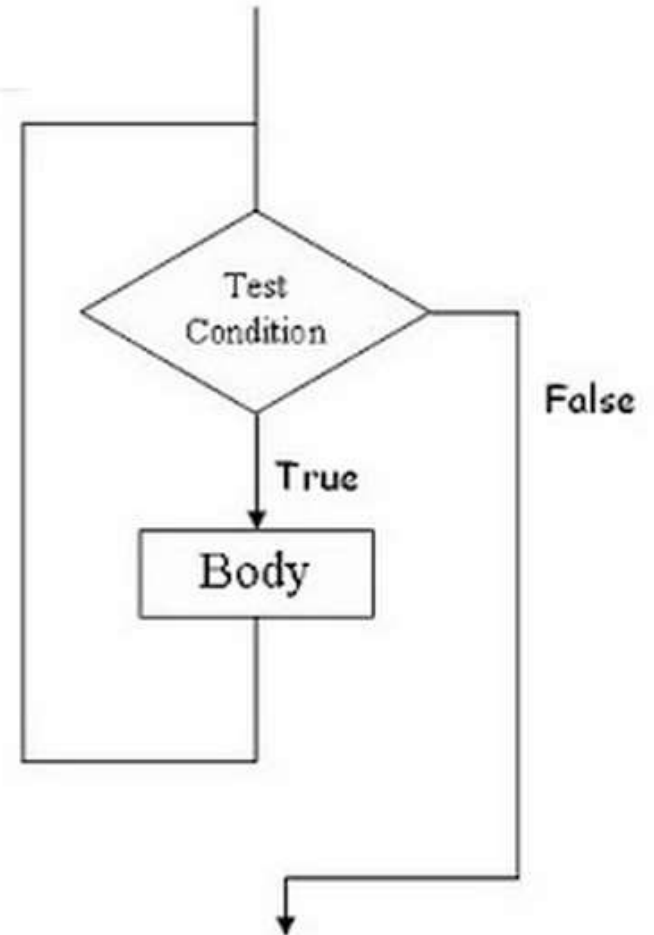


- **WHILE**
- **DO..WHILE**
- **FOR**
- **EXITING FROM A LOOP  
(BREAK, CONTINUE, GOTO)**

# While

## While Loop Syntax :

```
initialization;  
while (condition)  
{  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
incrementation;  
}
```



# While



- For **Single Line of Code** – Opening and Closing braces are not needed.
- **while(1)** is used for Infinite Loop
- Initialization , Incrementation and Condition steps are on different Line.
- While Loop is also **Entry Controlled Loop**. [i.e conditions are checked if found true then and then only code is executed ]

# While



```
File Edit Search Run Compile Debug Project Options Window Help
WHILE.C 1=1
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=0;
    clrscr();
    while(i<5)
    {
        printf(" %d\n",i);
        i++;
    }
    getch();
}
```

12:12

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

# While

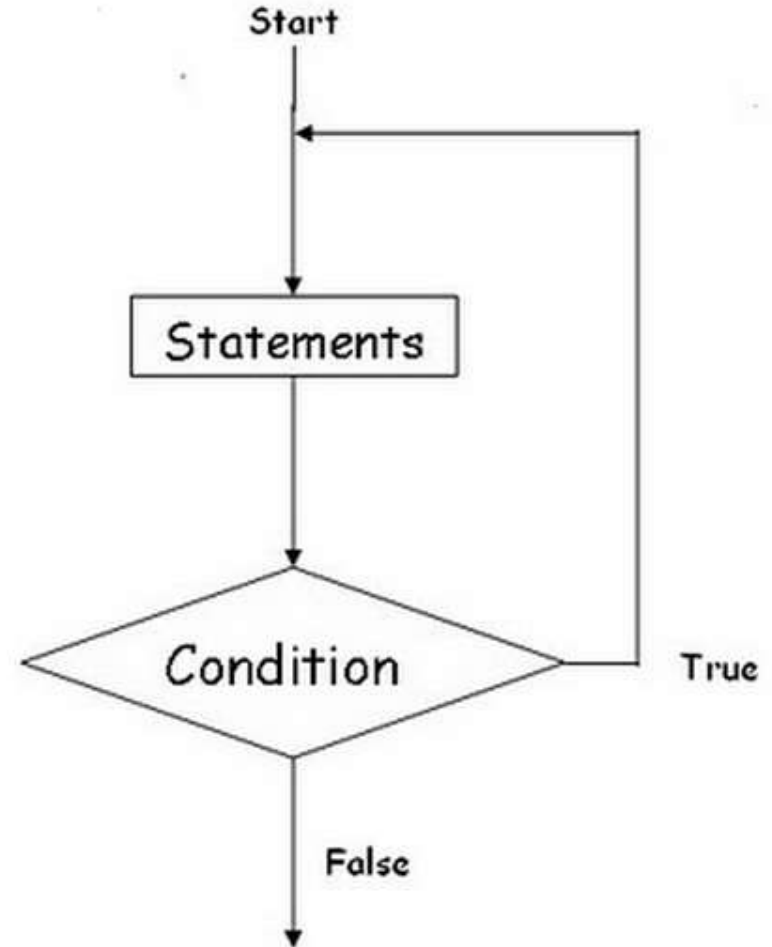


```
0  
1  
2  
3  
4
```

# Do...While

## Do-While Loop Syntax :

```
initialization;  
do  
{  
-----  
-----  
-----  
-----  
incrementation;  
}while(condition);
```



# Do...While



- It is **Exit Controlled Loop**.
- Initialization , Incrementation and Condition steps are on **different Line**.
- It is also called **Bottom Tested** [i.e Condition is tested at bottom and Body has to execute at least once ]

# Do...While



```
File Edit Search Run Compile Debug Project Options Window Help
DOWHILE.C 1=[↑]
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=0;
    clrscr();
    do
    {
        printf(" %d\n",i);
        i++;
    }while(i<5);
    getch();
}
```

11:5

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu



# Do...While



```
0  
1  
2  
3  
4
```

# For Loop



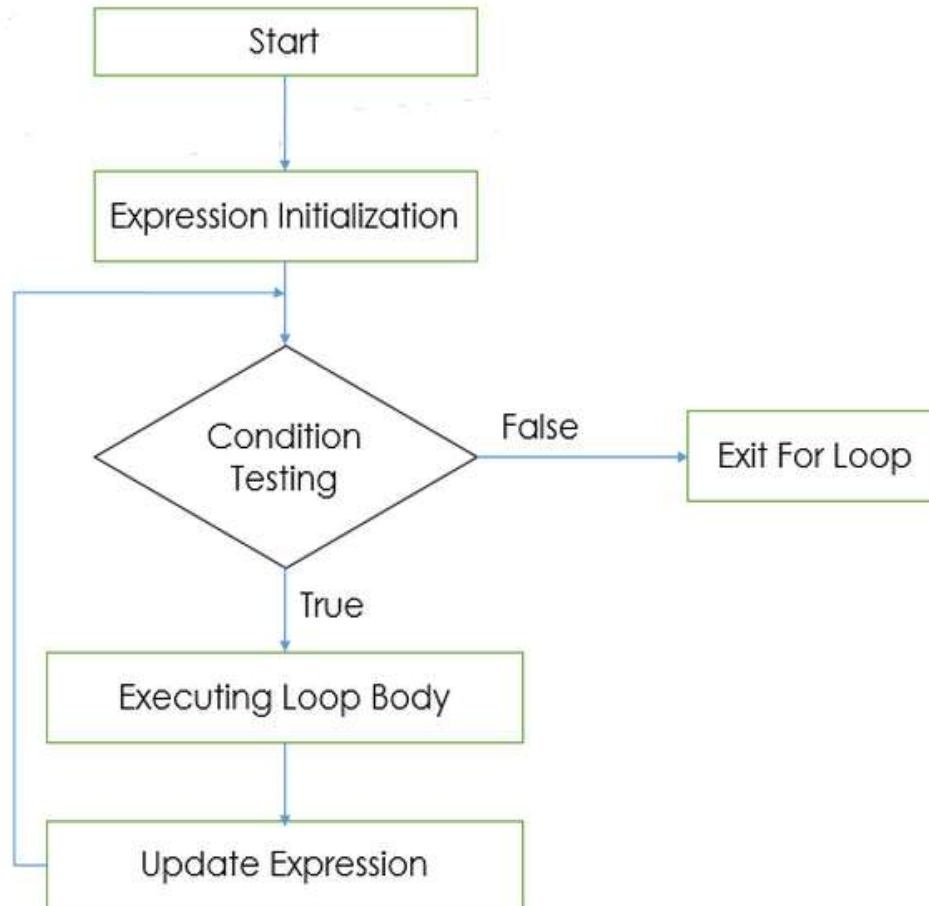
## Syntax of the For Loop :

```
for(initial expression; test expression; update expression)
{
body of loop;
}
```

# For Loop



## Flowchart of For Loop :



# For Loop



## **Explanation of For Loop :**

1. Firstly the for loop executes the initialize statement in which the subscript variable will be initialized with the initial value.
2. After the initialize statement the condition part of the loop will be executed if the condition becomes true then body of the loop will be executed otherwise the loop will be terminated
3. If the loop condition becomes true then body of the loop will be executed. After the execution of the for loop body the control goes to the third part of the loop statement i.e Expression Updation
4. After updating subscript variable control again goes to execute condition statement.

# For Loop



1. For **Single Line of Code** – Opening and Closing braces are not needed.
2. There can Exist **For Loop without body**.
3. Initialization , Incrementation and Condition steps are on same Line.
4. Like While loop , For Loop is **Entry Controlled Loop**. [i.e conditions are checked if found true then and then only code is executed ]

# For Loop



## Different Ways of Implementing For Loop

Form	Comment
<pre>for ( i=0 ; i &lt; 10 ; i++ ) Statement1;</pre>	<b>Single</b> Statement
<pre>for ( i=0 ; i &lt;10; i++) {     Statement1;     Statement2;     Statement3; }</pre>	<b>Multiple</b> Statements within for
<pre>for ( i=0 ; i &lt; 10;i++);</pre>	For Loop with no Body ( <b>Carefully Look at the Semicolon</b> )
<pre>for (i=0,j=0;i&lt;100;i++,j++) Statement1;</pre>	<b>Multiple initialization</b> & Multiple <b>Update</b> Statements Separated by Comma
<pre>for ( ; i&lt;10 ; i++)</pre>	<b>Initialization not used</b>
<pre>for ( ; i&lt;10 ; )</pre>	<b>Initialization &amp; Update not used</b>
<pre>for ( ; ; )</pre>	<b>Infinite</b> Loop, Never Terminates

# For Loop



```
File Edit Search Run Compile Debug Project Options Window Help
FOR.CPP 1=[↑]
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=0;
    clrscr();
    for(i=0;i<5;i++)
    {
        printf(" %d\n",i);
    }
    getch();
}
```

10:4

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

# For Loop



```
0  
1  
2  
3  
4
```



# Break Statement



- It is a jump instruction and can be used inside the switch and loop statements.
- The execution of the break statements causes the control transfer to the statement immediately after the loop.

# Break Statement

## Break in For Loop :

```
for(initialization ; condition ; increment  
ation)  
{  
Statement1;  
Statement2;  
break;  
}
```

## Break in While Loop :

```
initialization ;  
while(condition)  
{  
Statement1;  
Statement2;  
incrementation  
break;  
}
```

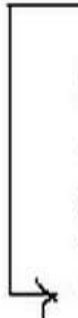
# Break Statement

## Break Statement in Do-While :

```
initialization ;  
do  
{  
Statement1;  
Statement2;  
incrementation  
break;  
}while(condition);
```

## Do-While Loop

```
do  
{  
-----  
-----  
if ( condition )  
    break ;  
-----  
-----  
} while ( condition )
```



# Break Statement



## Nested for

```
for ( - - - - )  
{  
  - - - -  
  - - - -  
    for ( - - - - )  
    {  
      - - - -  
      if ( condition )  
        break ;  
      - - - -  
    }  
  - - - -  
}
```

## For Loop

```
for ( - - - - )  
{  
  - - - -  
  - - - -  
  if ( condition )  
    break ;  
  - - - -  
  - - - -  
}
```

# Break Statement



## While Loop

```
while ( - - - - )
```

```
{
```

```
-----
```

```
-----
```

```
if ( condition )
```

```
break ;
```

```
-----
```

```
-----
```

```
}
```

```
-----
```



# Break Statement



The screenshot shows a Turbo C++ IDE window titled "BREAK.C". The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, Options, Window, and Help. The code in the editor is as follows:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=0;
    clrscr();
    for(i=0;i<5;i++)
    {
        printf(" %d\n",i);
        if(i==2)
            break;
    }
    getch();
}
```

The cursor is positioned at the end of the first line of the for loop. The status bar at the bottom shows the time "4:20" and keyboard shortcuts: F1 Help, Alt-F8 Next Msg, Alt-F7 Prev Msg, Alt-F9 Compile, F9 Make, and F10 Menu.

# Break Statement



```
0  
1  
2
```

# Continue Statement



- It is a jump statement.
- It is used only inside the loop.
- Its execution does not exit from the loop but escape the loop for that iteration and transfer the control back to the loop for the new iteration.



# Continue Statement

## Continue Statement :

```
loop
{
  continue;
  //code
}
```

- It is used for Skipping part of Loop.
- Continue causes the remaining code inside a loop block to be skipped and causes execution to jump to the top of the loop block

# Continue Statement



for

```
→ for ( initialization ; condition ; Iteration )  
{  
.....  
    if (---)  
        continue ;  
.....  
.....  
}
```

while

```
→ while ( condition )  
{  
.....  
    if (---)  
        continue ;  
.....  
.....  
}
```

# Continue Statement



```
do-while do {  
    .....  
    if (---)  
    {  
        continue ;  
    }  
    .....  
    .....  
} while (condition);
```

# Continue Statement



```
File Edit Search Run Compile Debug Project Options Window Help
CONTINUE.C
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=0;
    clrscr();
    for(i=0;i<5;i++)
    {
        if(i==2)
            continue;
        printf(" %d\n",i);
    }
    getch();
}
```

10:14

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

# Continue Statement



```
0  
1  
3  
4
```

```
-
```

# Goto Statement



- It is used to alter the normal sequence of flow by transferring the control to some other part of the program unconditionally.
- It is followed by the label statement which determines the instruction to be handled next after the execution of the goto statement.

# Goto Statement

## Goto Statement

```
goto label;
```

```
-----
```

```
-----
```

```
label :
```

Whenever goto keyword encountered then it causes the program to continue on the line , so long as it is in the scope .

## Types of Goto :

- Forward
- Backward

# Goto Statement



```
goto Label ;  
.....  
.....  
.....  
Label : ←
```

A diagram illustrating a forward goto statement. A horizontal line extends from the end of the 'goto Label ;' line to the right. A vertical line descends from the end of this horizontal line. A horizontal line then extends from the end of this vertical line to the left, ending with an arrowhead pointing to the 'Label :' line.

```
Label : ←  
.....  
.....  
.....  
goto Label ;
```

A diagram illustrating a backward goto statement. A horizontal line extends from the end of the 'goto Label ;' line to the right. A vertical line descends from the end of this horizontal line. A horizontal line then extends from the end of this vertical line to the left, ending with an arrowhead pointing to the 'Label :' line.



# Goto Statement



```
File Edit Search Run Compile Debug Project Options Window Help
GOTO.C
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=0;
    clrscr();
    for(i=0;i<5;i++)
    {
        if(i==2)
            goto next;
        printf(" %d\n",i);
    }
    next:
    printf("We are in the label");
    getch();
}
```

14:15

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

# Goto Statement



```
0  
1  
We are in the label_
```