

# Control Statements

Prof. Harish D.G.  
Dept. of Computer and IT  
College of Engineering, Pune  
[www.harishgadade.com](http://www.harishgadade.com)

# Control Statements

```
graph TD; A[Control Statements] --> B[Decision Making Statements]; A --> C[Looping / Iterative Statements]; A --> D[Jumping Statements]; B --> B1[- If - Statements]; B --> B2[- Switch-Case Statements]; B --> B3[- Conditional Statements]; C --> C1[- While Loop]; C --> C2[- Do-While Loop]; C --> C3[- For Loop]; D --> D1[- Break Statements]; D --> D2[- Continue Statements]; D --> D3[- Goto Statements];
```

## Decision Making Statements

- If - Statements
- Switch-Case Statements
- Conditional Statements

## Looping / Iterative Statements

- While Loop
- Do-While Loop
- For Loop

## Jumping Statements

- Break Statements
- Continue Statements
- Goto Statements

# Break Statement

- The break statement ends the loop immediately when it is encountered.
- **Syntax**

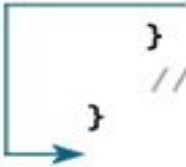
```
break;
```

- The break statement is almost always used with if...else statement inside the loop.

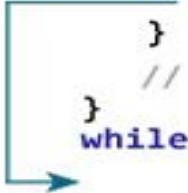
# Break Statement

- How break Statements Works?

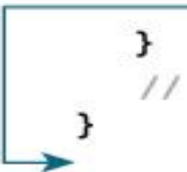
```
while (testExpression) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}
```



```
do {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
} while (testExpression);
```



```
for (init; testExpression; update) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}
```



# Break Statement

```
##include<stdio.h>
void main()
{
    int i;
    for(i=1;i<=10;i++)
    {

        if(i==3)
            break;
        printf("\nValue of i is %d",i);

    }
    printf("\nOutside of while loop\n");
}
```

# Break Statement

```
##include<stdio.h>
void main()
{
    int i;
    for(i=1;i<=10;i++)
    {

        if(i==3)
            break;
        printf("\nValue of i is %d",i);
    }
    printf("\nOutside of while loop\n");
}
```

## Output:

```
Value of i is 1
Value of i is 2
Outside of while loop
```

# Continue Statement

- The `continue` statement skips the current iteration of the loop and continues with the next iteration.

- **Syntax**

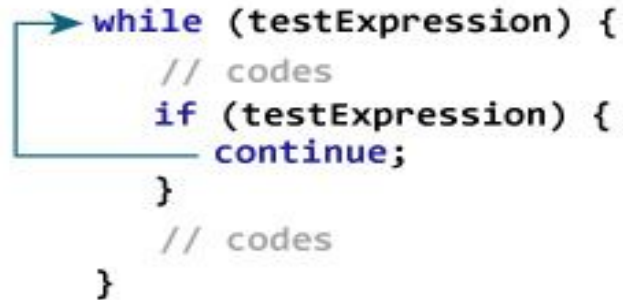
```
continue;
```

- The `continue` statement is almost always used with the `if...else` statement.

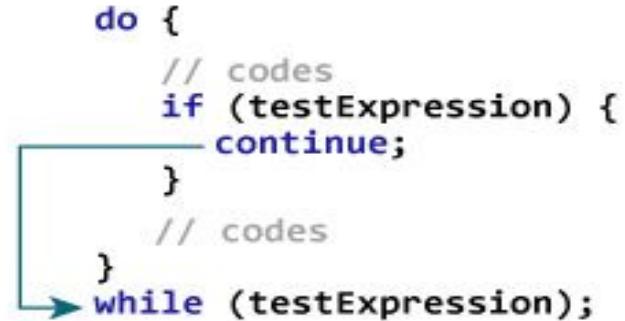
# Continue Statement

- How continue Statement works?

```
while (testExpression) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```



```
do {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
} while (testExpression);
```



```
for (init; testExpression; update) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```





# Continue Statement

- **Example**

```
#include<stdio.h>
void main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==3)
            continue;
        printf("\nValue of i is %d",i);
    }
    printf("\nOutside of while loop\n");
}
```

# Continue Statement

- **Example**

```
#include<stdio.h>
void main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==3)
            continue;
        printf("\nValue of i is %d",i);
    }
    printf("\nOutside of while loop\n");
}
```

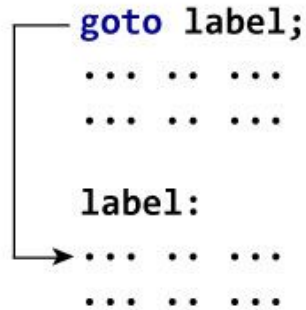
## Output:

```
Value of i is 1
Value of i is 2
Value of i is 4
Value of i is 5
Value of i is 6
Value of i is 7
Value of i is 8
Value of i is 9
Value of i is 10
Outside of while loop
```

# goto Statement

- The `goto` statement allows us to transfer control of the program to the specified label.
- **Syntax**

```
goto label;  
... ..  
... ..  
  
label:  
... ..  
... ..
```

A diagram illustrating the execution of a goto statement. It shows a code snippet with a 'goto label;' statement at the top, followed by two lines of ellipses representing code. Below this is a label 'label:' followed by two more lines of ellipses. A line starts from the end of the 'goto' statement, goes down, then left, then down again, and finally right as an arrow pointing to the first line of code under the 'label:'.

- The label is an identifier. When the goto statement is encountered, the control of the program jumps to label: and starts executing the code.

# goto Statement

```
#include<stdio.h>
void main()
{
    int n=10;
    if(n%2==0)
        goto even;
    else
        goto odd;

    even:
        printf("\n%d is Even Number\n",n);

    odd:
        printf("\n%d is Odd Number\n",n);
}
```

# goto Statement

```
#include<stdio.h>
void main()
{
    int n=10;
    if(n%2==0)
        goto even;
    else
        goto odd;

    even:
        printf("\n%d is Even Number\n",n);

    odd:
        printf("\n%d is Odd Number\n",n);
}
```

**Output:**

10 is Even Number