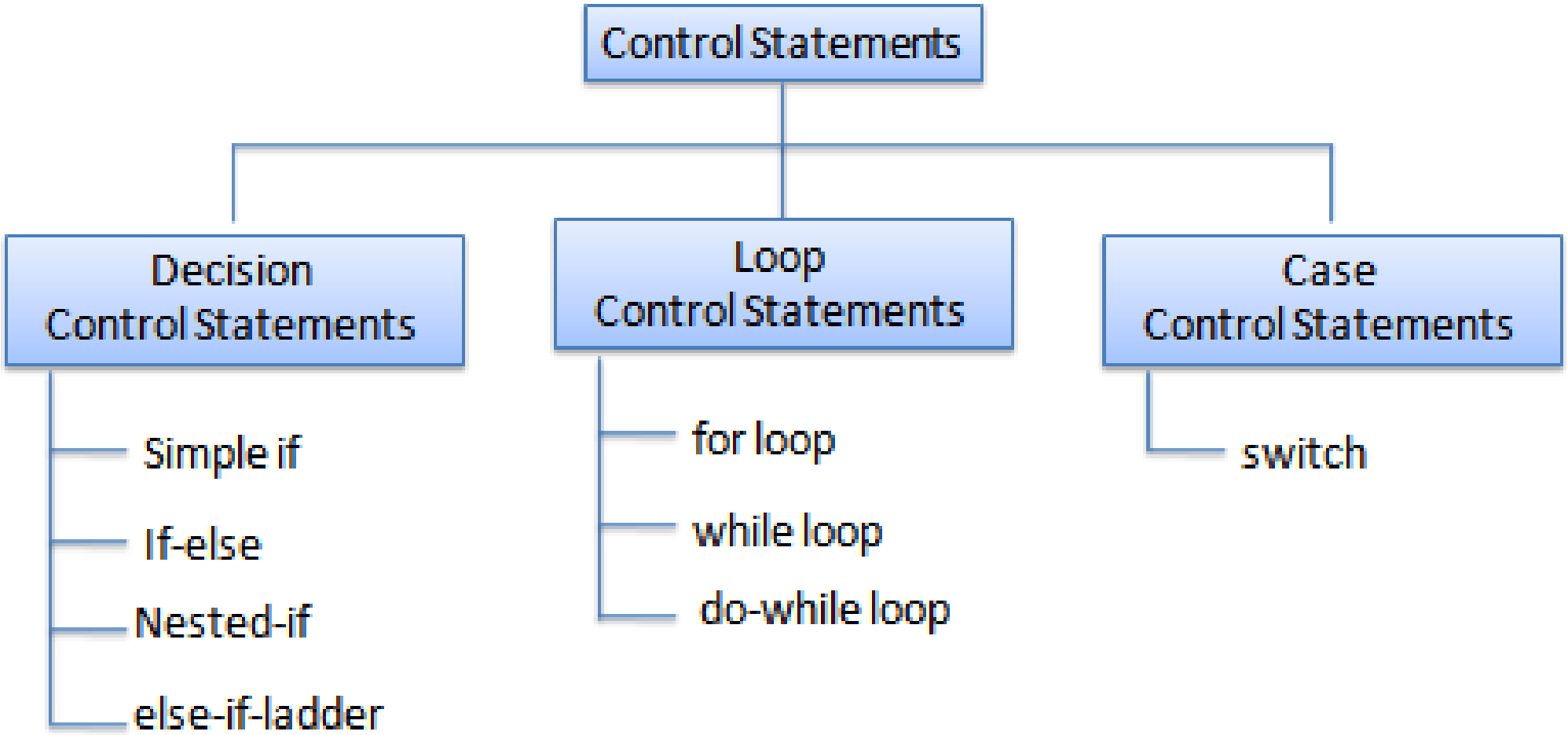
C Programming

UNIT 2 Handouts / Class Notes

Dr. Suresh Mudunuri, SRKR Engineering College

UNIT 2 Syllabus

UNIT-II (10 Hrs)	Bitwise Operators: Exact Size Integer Types, Logical Bitwise Operators, Shift Operators. Selection & Making Decisions: Logical Data and Operators, Two Way Selection, Multiway Selection, More Standard Functions. Repetition: Concept of Loop, Pretest and Post-test Loops, Initialization and Updating, Event and Counter Controlled Loops, Loops in C, Other Statements Related to Looping, Looping Applications, Programming Examples.



simple if:

```
if (condition)
{
statement 1;
statement 2;
.
.
.
.
statement n;
}
```

These statements get executed only when the "if" condition is true, otherwise the control comes out of the "if" statement.

if-else:

```
if (condition)
{
statement 1;
}
else
{
statement 2;
}
```

this statement gets executed when "if" condition is true.

this statement gets executed when "if" condition fails and control shifts to "else".

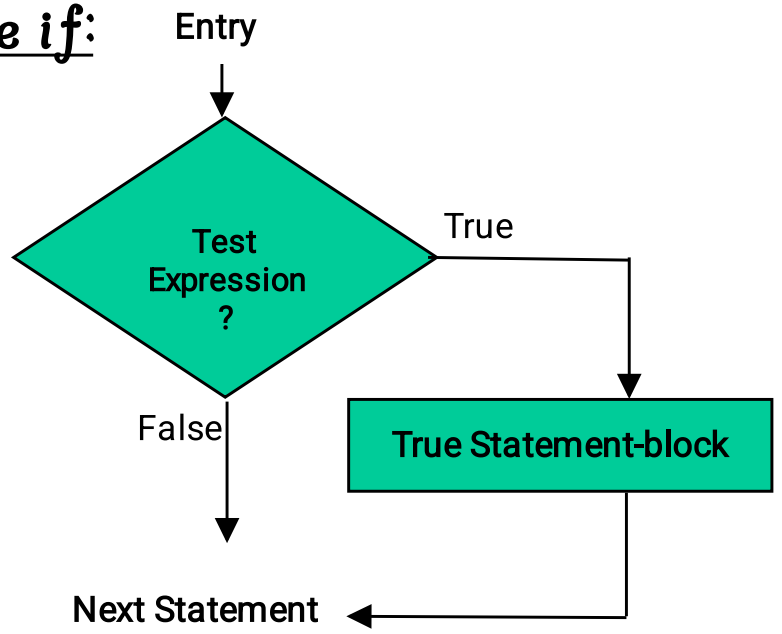
If-else ladder:

```
if(condition 1)
{
statement 1;
}
else if(condition 2)
{
statement 2;
}
else if(condition3)
{
statement 3;
}
.
.
.
else
{
statement;
}
```

In this different test conditions are verified and the statements are executed when the conditions are true.

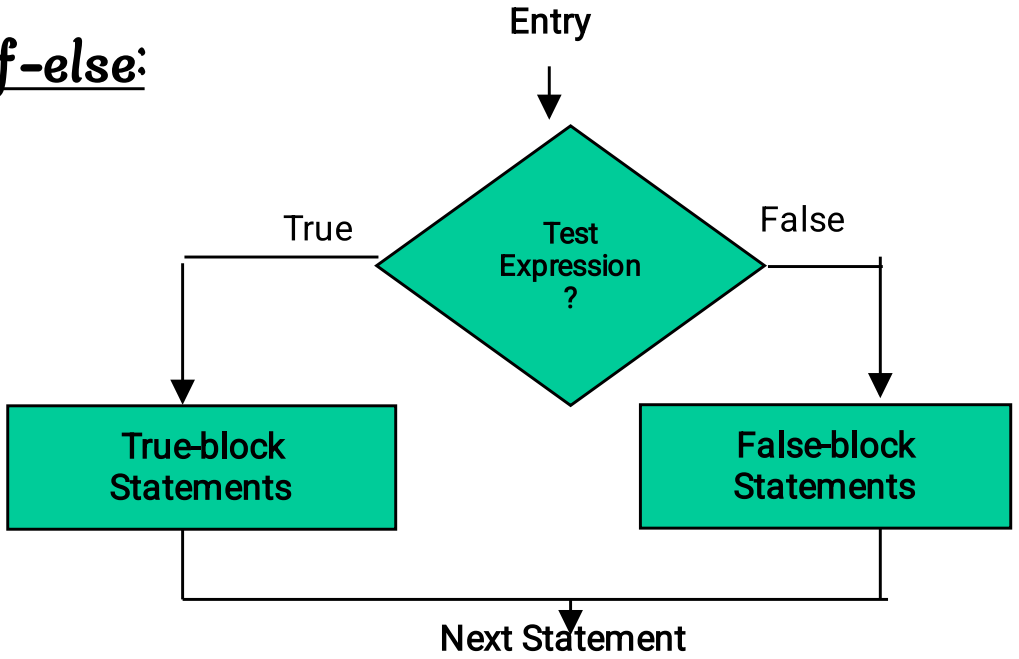
This statement is executed when all the "if" and "else if" conditions fail.

simple if:



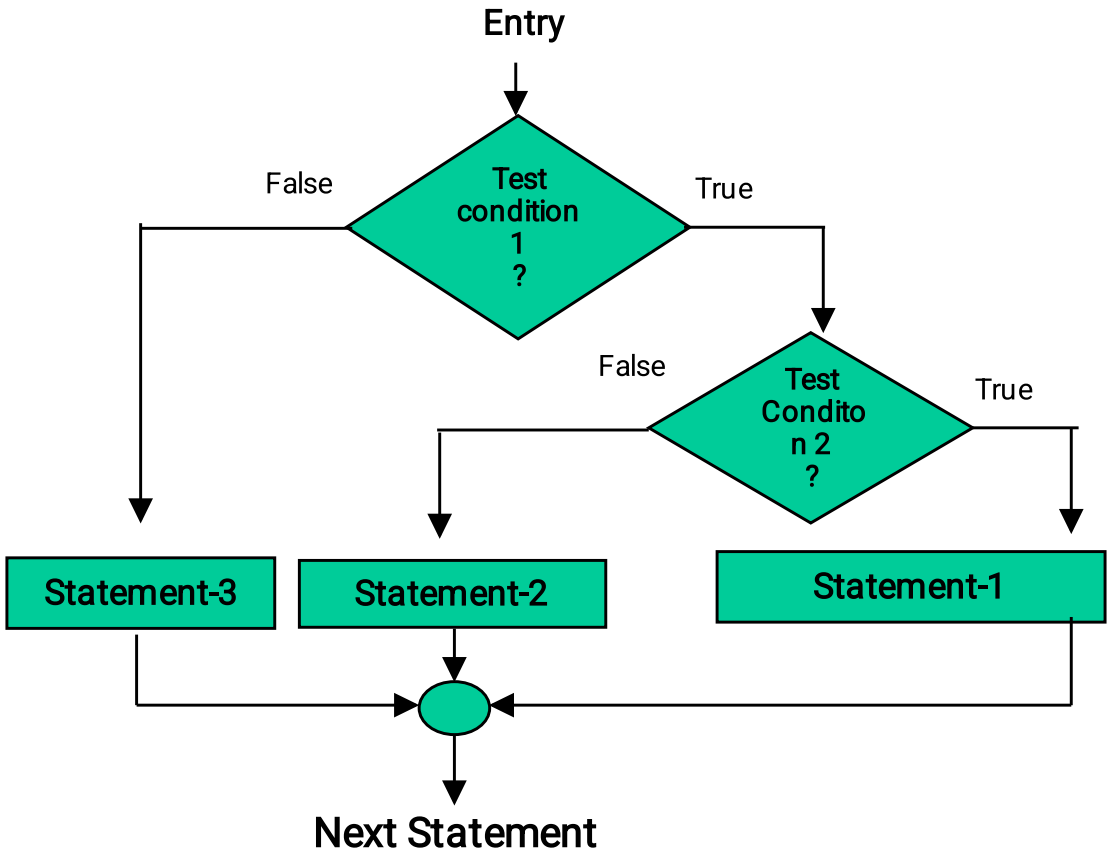

```
/* check a citizen is eligible for voting */

#include<stdio.h>
int main()
{
    int age;
    printf("Enter the age : ");
    scanf("%d",&age);

    if(age >= 18)
       printf("Eligible for voting...");

    return 0;
}
```

if-else:



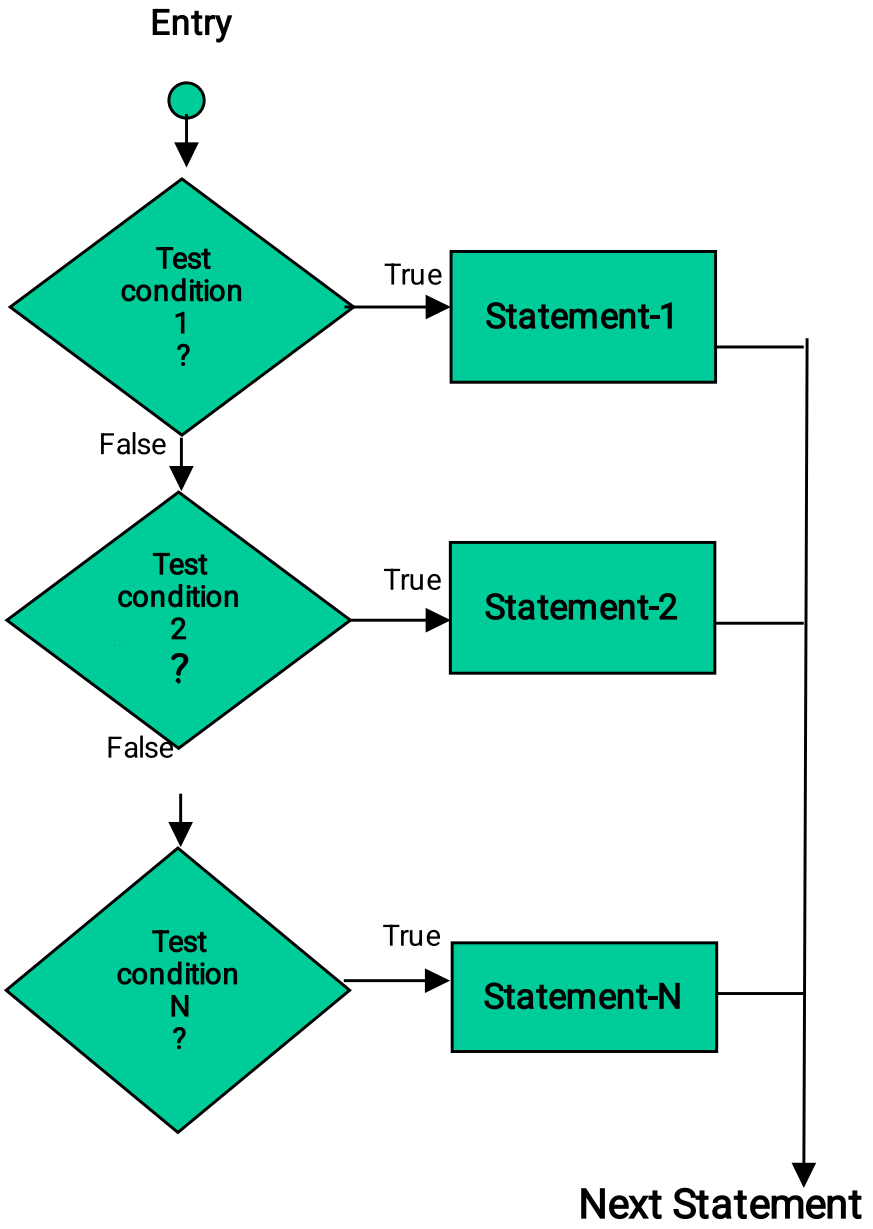

```
/* print a number is even or odd */

#include<stdio.h>
int main()
{
     int number;
     printf("Enter a number : ");
     scanf("%d", &number);

     if((number %2) == 0)
        printf("%d is even number.",number);
     else
        printf("%d is odd number.",number);
}
```

nested if...else:



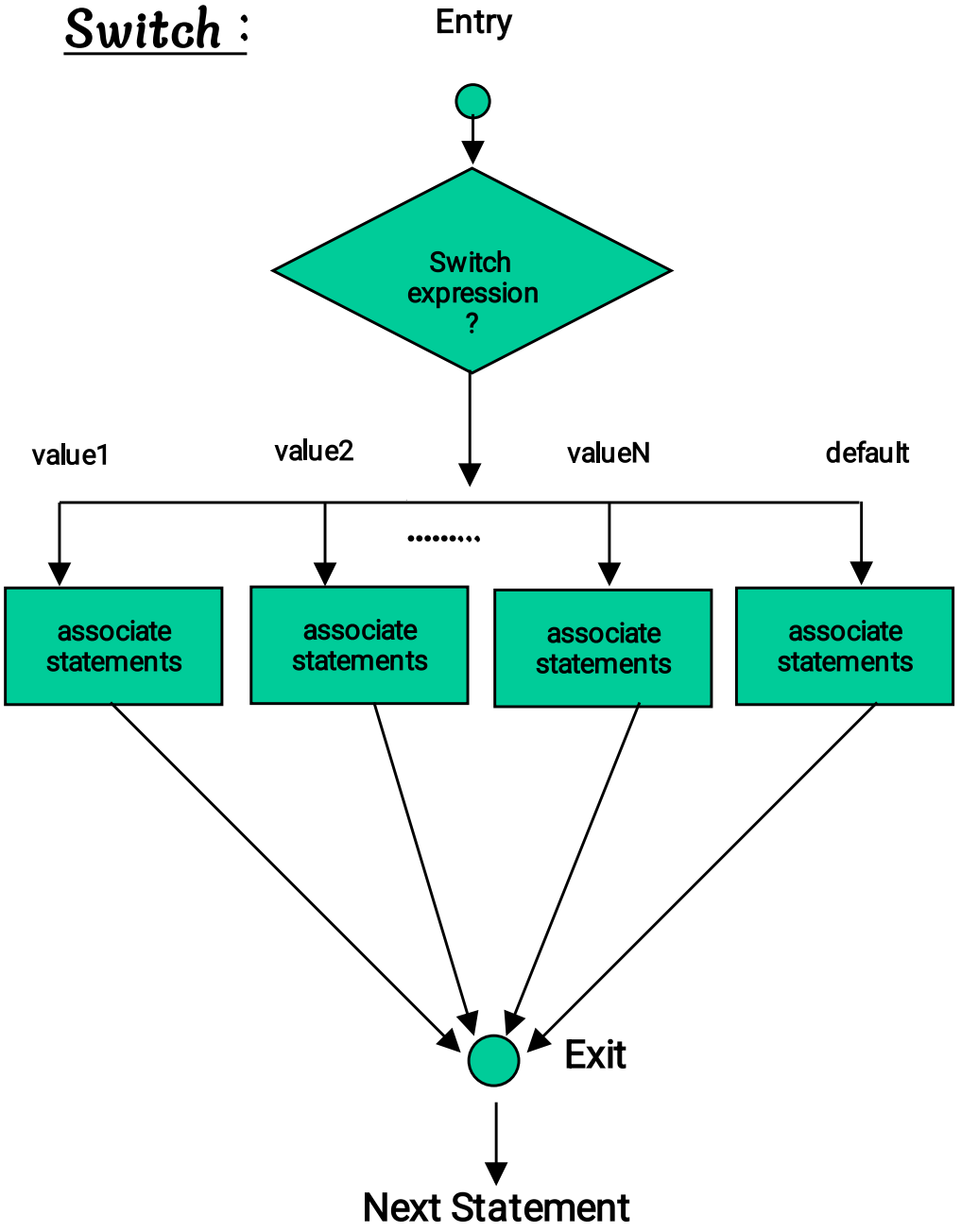

```
/* check whether a year is leap year or not */

#include<stdio.h>

int main()
 {
    int year;
    printf("Enter the year ?");
    scanf("%d",&year);

    if((year %100) == 0)
     {
          if((year % 400) == 0)
             printf("%d is leap year.",year);
          else
             printf("%d is not leap year.",year);

     }
   else
     {
          if((year % 4) == 0)
             printf("%d is leap year.",year);
          else
             printf("%d is not leap year.",year);
     }
    return 0;
}
```

if...else...if :




```c
/* program to print the grade of student */

#include<stdio.h>

int main()
{
    int marks;
    printf("Enter marks ? ");
    scanf("%d", &marks);

   if(marks >= 75)
       printf("Distinction");
    else if(marks >= 60)
       printf("First class");
    else if(marks >= 50)
       printf("Second class");
    else if(marks >= 35)
       printf("Third class");
    else
       printf("Failed");

   return 0;
}
```

Switch :



Syntax

```
switch(var)
{
case var1:  statement;
            break;
case var2: statement;
            break;
case var3: statement;
            break;
.
.
.
default:    statement;
}
```

This statement gets executed when the entered variable doesn't match with any.

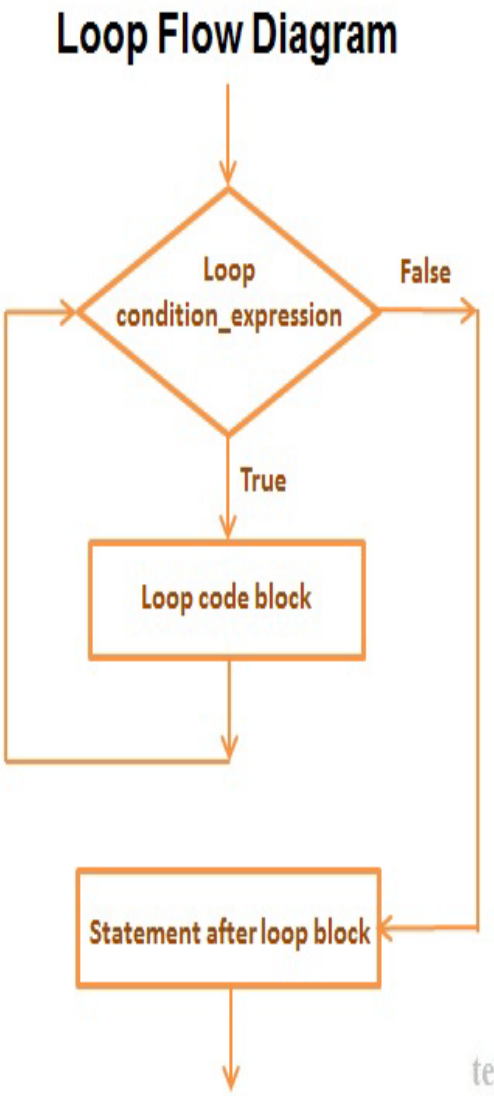
switch statement :

```c
void main()
{
int roll = 3 ;
switch ( roll )
      {
      case 1 :
            printf("I am Pankaj");
            break;
      case 2 :
            printf("I am Nikhil");
            break;
      case 3 :
            printf("I am John");
            break;
      default :
            printf("No student found");
            break;
       }
}
```

```c
/* program to simulate a simple calculator */

#include<stdio.h>
void main() {
    float a,b;
    char opr;
    printf("Enter number1 operator number2 : ");
    scanf("%f %c %f",&a,&opr,&b);

    switch(opr)
    {
        case '+':
            printf("Sum : %f",(a + b));
            break;
        case '-':
            printf("Difference : %f",(a - b));
            break;
        case '*':
            printf("Product : %f",(a * b));
            break;
        case '/':
            printf("Quotient : %f",(a / b));
            break;
        default:
            printf("Invalid Operation!");
    }
}
```

switch(op) → Op should be either an integer or arithmetic expression that results in an integer or a character constant or variable

Loop Statements



DIFFERENT TYPES OF LOOP

Loop Type	Description
While loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
For loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
Do.....While loop	Like a while statement, except that it tests the condition at the end of the loop body
Nested loop	You can use one or more loop inside any another while, for or do..while loop.

While Loop (Entry Controlled Loop)

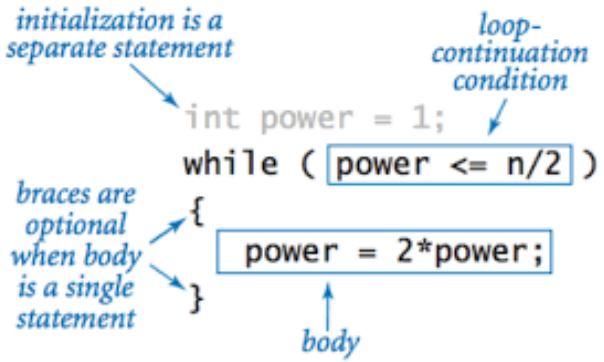
while loop

Syntax

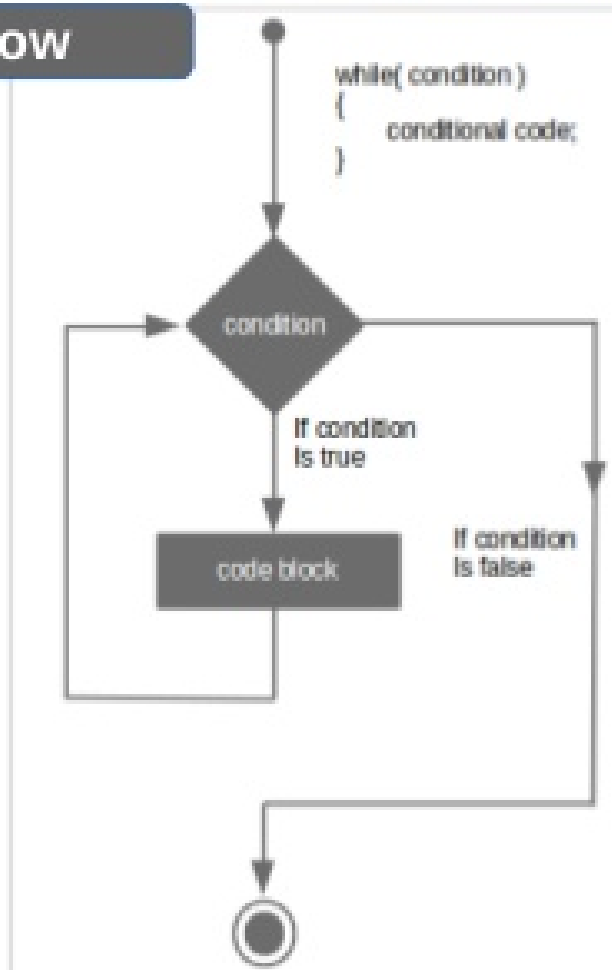
```
1 while(condition)
2 {
3         //body
4 }
5
```

Example

```
1 int i = 1;
2
3 while(i < 10)
4 {
5         printf("%d", i);
6         i++;
7 }
8
```



Flow



Do-While Loop (Exit Controlled Loop)

do-while loop

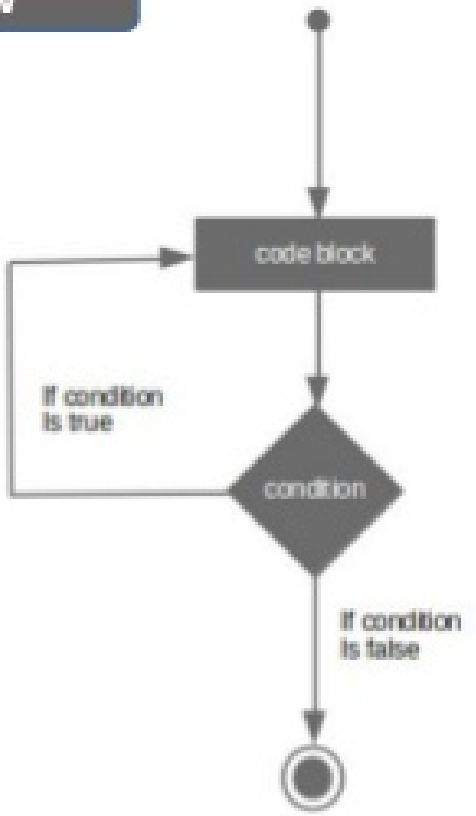
Syntax

```
1 do
2 {
3         //body
4 }while(condition);
5
6
```

Example

```
1 int i = 1;
2 do
3 {
4         printf("%d", i);
5         i++;
6 }while(i < 10);
7
```

Flow



For Loop

for loop

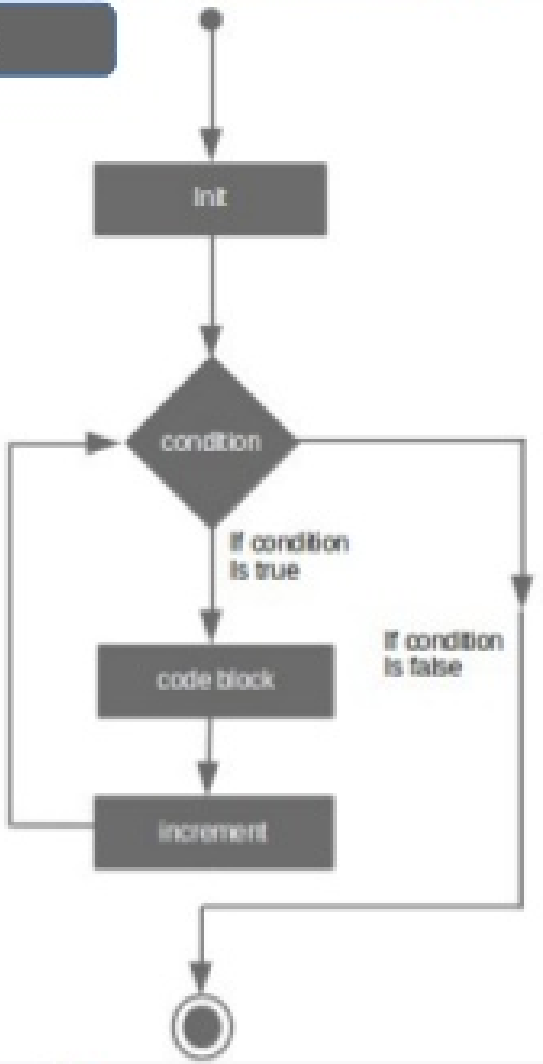
Syntax

```
1 for(init; condition; increment)
2 {
3         //body
4 }
5
```

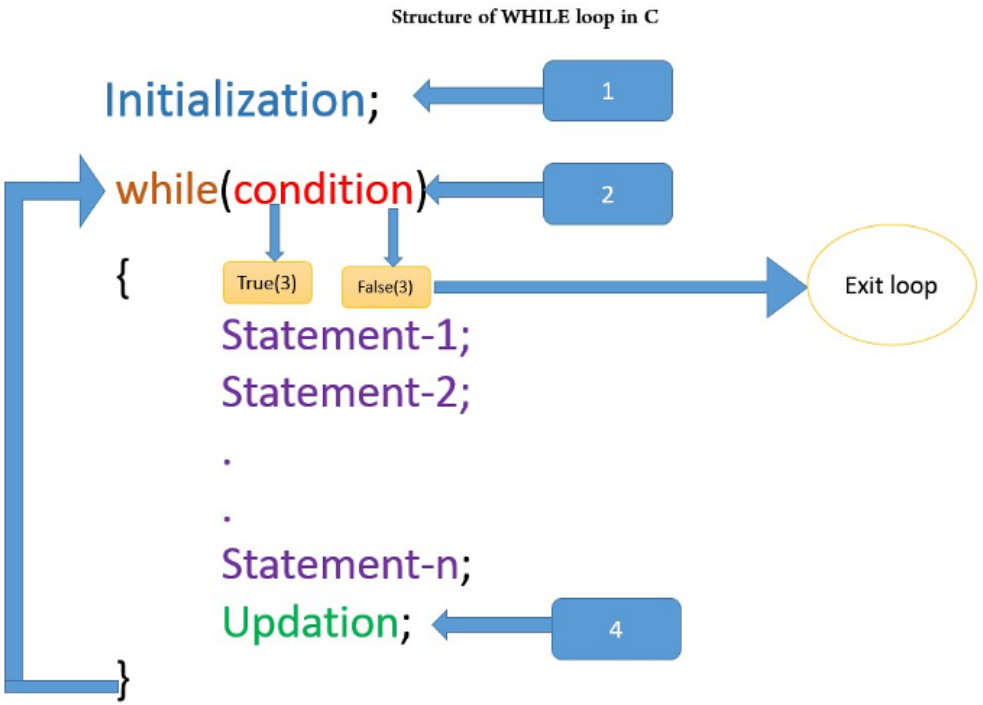
Example

```
1 int i;
2 for(i = 0; i < 10; i++)
3 {
4         printf("%d", i);
5 }
6
```

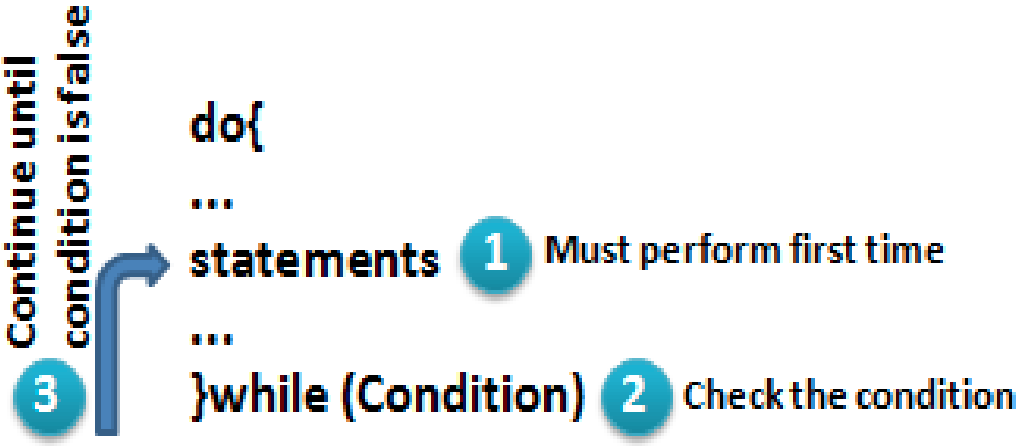
Flow



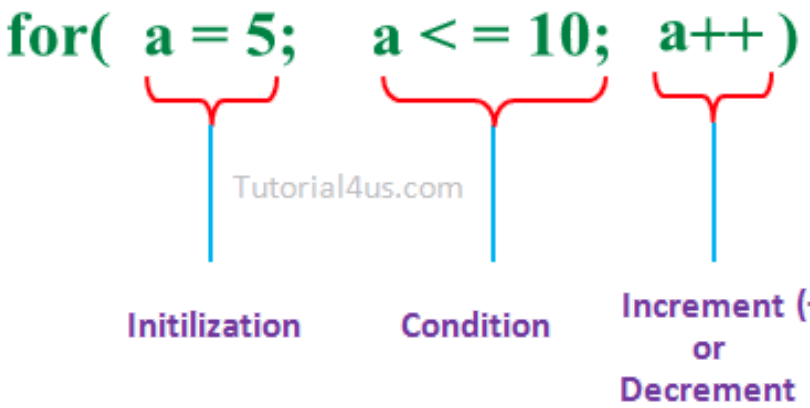
While - Flow of Control

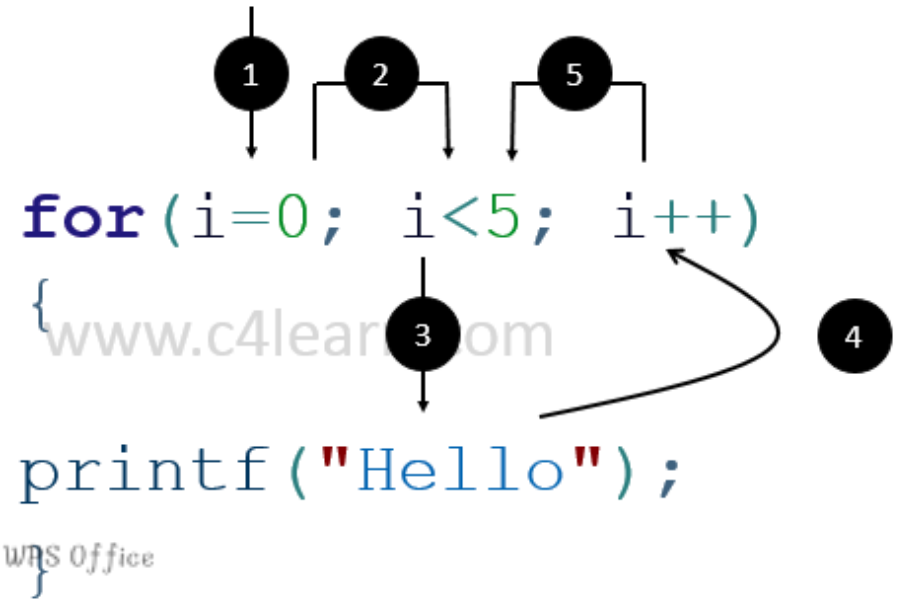


Do-While - Flow of Control



For - Flow of Control





Loops Summary

	When to use	Template

■ for loop:	If you know, prior to the start of loop, how many times you want to repeat the loop.	`for (int i=0; i<max; i++)`
		`{`
		`  <statement(s)>`
		`}`

■ do loop:	If you always need to do the repeated thing at least one time.	`do`
		`{`
		`  <statement(s)>`
		`  <prompt - do it again (y/n)?>`
		`} while (<response == 'y'>);`

■ while loop:	If you can't use a `for` loop or a `do` loop.	`<prompt - do it (y/n)?>`
		`while (<response == 'y'>)`
		`{`
		`  <statement(s)>`
		`  <prompt - do it again (y/n)?>`
		`}`

Loops - Sample Programs

While Loop

```
/* print multiples of 5 till 100*/

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

Do-While Loop

```
/* average of 5 numbers */

#include<stdio.h>
int main() {
    int count = 1;
    float x,  sum = 0;
    do {
       printf("x = ");
       scanf("%f",&x);
       sum += x;
       ++count;
    }  while(count <= 5);
    printf("Average = %f ", (sum/5));
}
```

For Loop

```
/* check whether a number is prime or not */

#include<stdio.h>
int main()
 {
   int n,i,factors = 0;

   printf("Enter a number : ");
   scanf("%d",&n);

   for(i = 1; i <= n; i++) {
     if((n % i)==0)  ++factors;
   }

   if (factors == 2)
     printf("%d is prime number.",n);
   else
     printf("%d is not prime number.",n);

}
```

Nesting of Loops

- ***NESTED LOOPS :***
- A loop can be nested inside of another loop.
- **Syntax:**
  The syntax for a **nested for loop** statement in C is as follows:

```
for ( init; condition; increment )
  {
     for ( init; condition; increment )
        {
           statement(s);
         }
     statement(s);
     // you can put more statements.
  }
```

```
int i, j;
for (i=1; i<=6; i++)
{
     for (j=1; j<=i; j++)
          {
          printf("*");
          }
     printf("\n");
}
```

Event Controlled Vs Counter Controlled Loops

- All the possible expressions that can be used in a loop limit test can be summarized into two general categories:
  1. **Event-controlled loops and**
  2. **Counter-controlled loops.**
- In event-controlled loops, loop execution depends on the given condition.
- In counter-controlled loops, loop execution depends on the counter variable value.

break, continue Statement

- **break** statement is used to jump out of the loop instantly.

e.g.

```
for(i = 0; i <= 10; i++)
{
        if(i == 5)
                break;
        printf("%d  ", i);
}
```

The output of the above program is:

0 1 2 3 4

- **continue** statement takes the control of the program to the beginning of loop.

e.g.

```
for(i = 0; i <= 10; i++)
{
        if(i == 5)
                continue;
        printf("%d  ", i);
}
```

The output of the above program is:

0 1 2 3 4 6 7 8 9 10