Decision Making and Branching

Conditional Branching	Conditional Branching
→if statement →nested if statement →if ..else statement →nested if else statement	→break →continue →goto

These include conditional type branching and unconditional type branching.

if statement

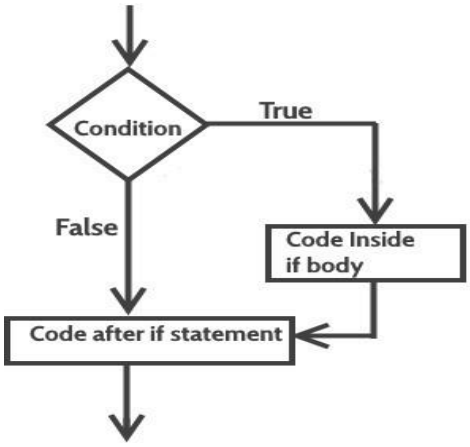
It takes the following form

```
if(test-expression)
```

It allows the computer to evaluate the expression first and them depending on whether the value of the expression is "true" or "false", it transfer the control to a particular statement. This point of program has two paths to flow, one for the true and the other for the false condition.

```
if(test-expression)
{
statement-block
}
statement-x;
```

The statement-block may be a single statement or group of statements. If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x. But when is condition true both the statement-block and the statement-x are executed in sequence.



Eg-2

```
if (code = = 1)
{
salary = salary + 500;
}
printf("%d",salary);
```

Eg: Example program: C Program to check equivalence of two numbers using if statement

```
#include<stdio.h>
void main()
{
int m,n;
clrscr();
printf(" \n enter two numbers:");
scanf(" %d %d", &m, &n);
if(m-n= = 0)
{
printf(" \n two numbers are equal");
}
getch();
}
```

o/p

```
4 4

two numbers are equal
```

Nested if

The syntax for a nested if statement is as follows

```
if( cond 1)
{
            /* Executes boolean expression when cond 1 is true */
   if(cond 2) {
             /* Executes when the boolean expression 2 is true */
   }
}
```

Example:

```
#include <stdio.h>
 int main ()
{
   int a = 100;
   int b = 200;
       if( a == 100 ) {
                          /* if condition is true then check the following */
      if( b == 200 ) {
                          /* if condition is true then print the following */

         printf("Value of a is 100 and b is 200\n" );
      }
   }
      printf("Exact value of a is : %d\n", a );
   printf("Exact value of b is : %d\n", b );
    return 0;
}
```

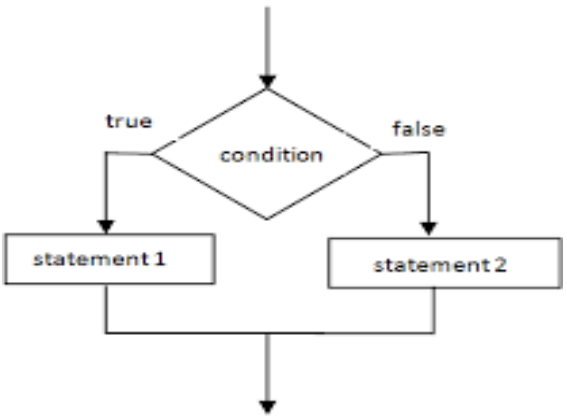
The if-else statement

The if-else statement is an extension of the simple if statement. The general form is If the test-expression is true, then true-block statements immediately following if statement are executed otherwise the false-block statements are executed.

```
if(test-expression)
{true-block statements
}
else
{
false-block statements
}statement-x
```

Example: C program to find largest of two numbers

```
#include<stdio.h>
int main()
{
int m,n,large;
printf(" \n enter two numbers:");
scanf(" %d %d", &m, &n);
if(m>n)
large=m;
else
large=n;
printf(" \n large number is = %d", large);
return 0;
}
```



Nested if-else statement

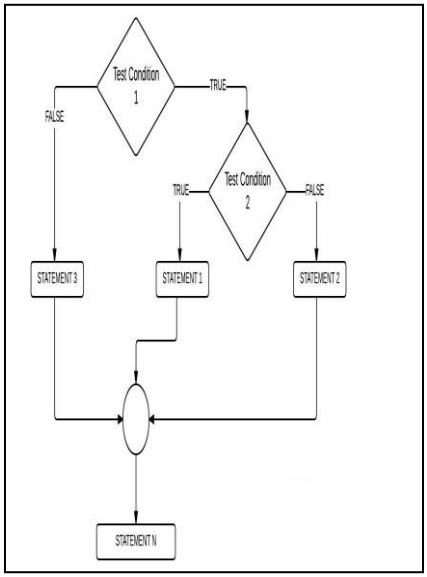
Nested if construct is also known as **if-else-if** construct.

Syntax-1

```
if(test-condition-1)
{
   (stmts)
else
  {
       if(condition 2)
      {
        Statement-1;
      }
   else
     {
       statement-2;
     }
}
}
  statement-x
```

Syntax-2

```
If(test-condition-1)
 {
   if(test-condition-2)
  {
          statement-1;
  }
   else
     {
        statement-2;
     }
}
else
{
statement-3;
}
   statement-x
```



If the test-condition-1 is false, the statement-3 will be executed; other wise it continues the second test. If the condition-2 is true, the statement-2 will be evaluated and then the control is transferred to the statement-x.

Example:Program to relate two integers using =, > or <

```
#include <stdio.h>
int main()
{
    int  number1,  number2;
    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

                  //checks if two integers are equal.
    if(number1 == number2)
    {
        printf("Result: %d = %d",number1,number2);
    }

                   //checks if number1 is greater than number2.
    else if (number1 > number2)
    {
        printf("Result: %d > %d", number1, number2);
    }

                  // if both test expression is false
    else
    {
        printf("Result: %d < %d",number1, number2);
    }

    return 0;
}
```

The switch statement:

When many conditions are to be checked then using nested if...else is very difficult, confusing and cumbersome.So C has another useful built in decision making statement known as switch.

This statement can be used as multiway decision statement.The switch statement tests the value of a given variable or expression against a list of case values and when a match is found, a block of statements associated with that case is executed.

Eg-1

```
int i = 1;
switch(i)
{
case 1:
        printf("A");
         break;
case 2:
         printf("B");
         break;
case 3:
         printf("C");
         break;
default:
 }
```

```
switch( code)
{
case 1:
        stmts1;
         break;
case 2:
         stmts2;
        break;
case 3:
         stmts3
         break;
 default:
         stmtsx
}
```

Example2: C program to find largest of two numbers

```
#include<stdio.h>
void main ()
{
float basic , da , salary ;
int code ;
char name[25];
da=0.0;
printf("Enter employee name\n");
scanf("%[^\n]",name);
printf("Enter basic salary\n");
scanf("%f",&basic);
printf("Enter code of the Employee\n");
scanf("%d",&code);
switch (code)
{case 1:
da = basic * 0.10;
break;
case 2:
da = basic * 0.15;
break;
case 3:
da = basic * 0.20; break;
default :
da = 0;
} salary = basic + da; printf("Employee name is\n");printf("%s\n",name);
printf ("DA is %f and Total salary is =%f\n",da, salary);
getch();
}
```

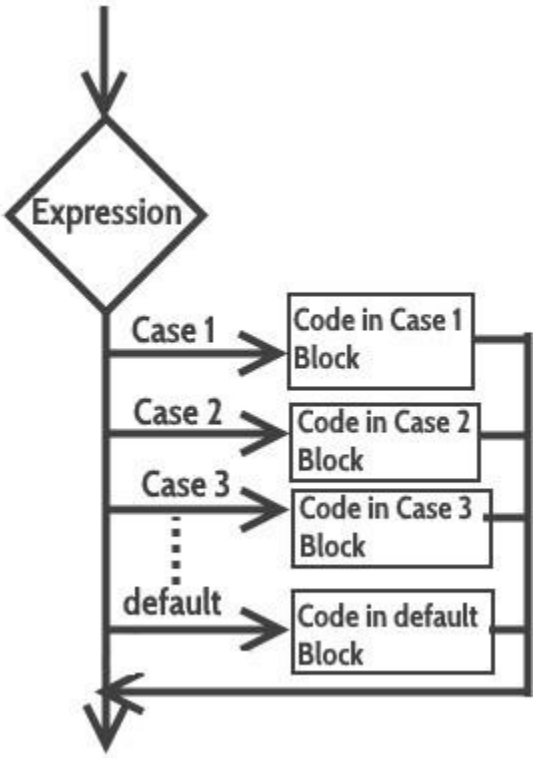
For case 1,da=10% of basic salary.
For case 2, da=15% of basic salary.
For case 3, da=20% of basic salary.
For default case >3 da is not given.(da=0)

o/p

```
Enter name of employee:
Kartiyani
Enter Basic salary
5000

Enter code of employee
1

Employee name is
Kartiyani
DA is 500 and total salary is 5500
```



Rules for using `switch` statement

1. The expression (after switch keyword) must yield an **integer** value
2. The case **label** values must be unique.
3. The case label must end with a colon(**:**)

Difference between `switch` and `if`

- `if` statements can evaluate `float` conditions. `switch` statements cannot evaluate `float` conditions.
- `if` statement can evaluate relational operators. `switch` statement cannot evaluate relational operators i.e they are not allowed in `switch` statement.

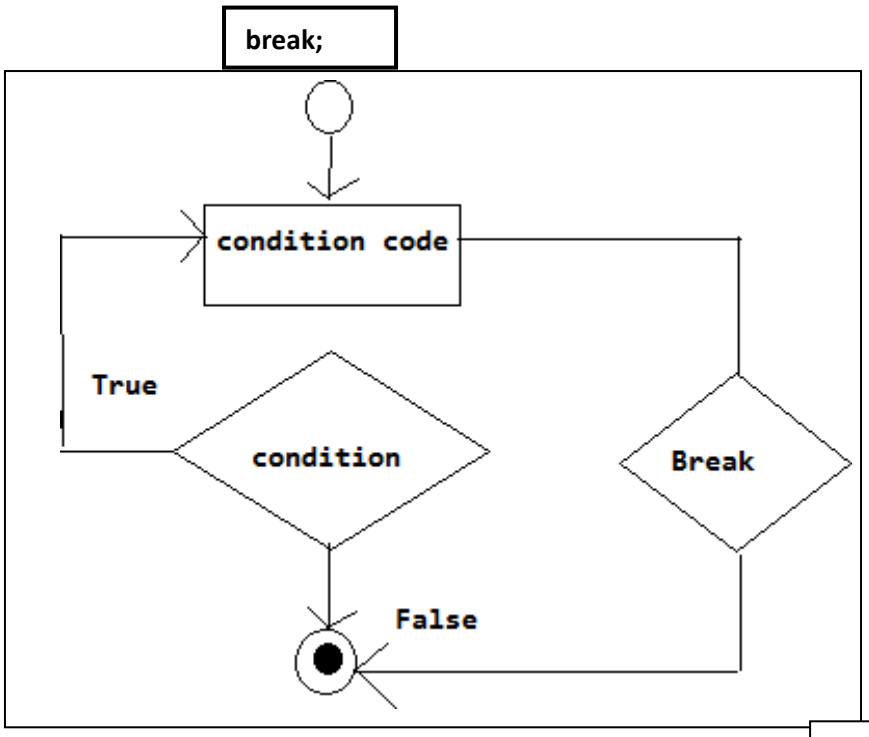
BREAK

The break statement in C programming has the following two usages −

- **When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.**
- **It can be used to terminate a case in the switch statement**

BREAK is a keyword that allows us to jump out of a loop instantly, without waiting to get back to the conditional test.

The syntax for a break statement in C is as follows −



Example program

```
#include <stdio.h>
int main ()
{
    int a = 10;
  while( a < 20 )
     {

        printf("value of a: %d\n", a);
        a++;
      if( a > 15)
       {
        break;
       }
    }
   return 0;
}
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

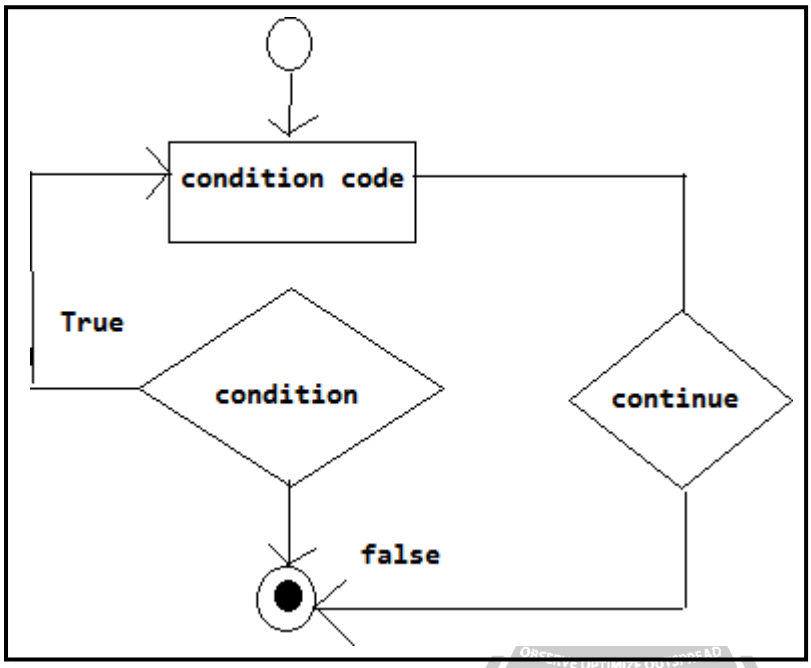
Continue

The continue statement in C programming works somewhat like the break statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.

For the for loop, continue statement causes the conditional test and increment portions of the loop to execute. For the while and do...while loops, continue statement causes the program control to pass to the conditional tests.

Syntax:

```
continue;
```



Example program

```
#include <stdio.h>
 int main ()
 {
      int a = 10;
    do {
         if( a == 15)
         {
           /* skip the iteration */
         a = a + 1;
         continue;
         }
       printf("value of a: %d\n", a);
       a++;
       } while( a < 20 );
   return 0;
}
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Break	Continue
The `break` statement can be used in both `switch` and loop (`for`, `while`, `do while`) statements.	**The `continue` statement can appear only in loops. You will get an error if this appears in switch statement.**
A `break` causes the `switch` or loop statements to terminate the moment it is executed. Loop or `switch` ends abruptly when break is encountered.	**A `continue` doesn't terminate the loop, it causes the loop to go to the next iteration. The `continue` statement is used to skip statements in the loop that appear after the `continue`.**
The `break` statement can be used in both `switch` and loop statements.	**The `continue` statement can appear only in loops. You will get an error if this appears in switch statement.**
When a `break` statement is encountered, it terminates the block and gets the control out of the `switch` or loop.	**When a `continue` statement is encountered, it gets the control to the next iteration of the loop.**

GOTO

GOTO STATEMENT

'C' supports goto statement to branch unconditionally from one point to another in the program.

A **goto** statement in C programming provides an unconditional jump from the 'goto' to a labeled statement.

NOTE − Use of **goto** statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten to avoid them.

Syntax

The syntax for a **goto** statement in C is as follows −

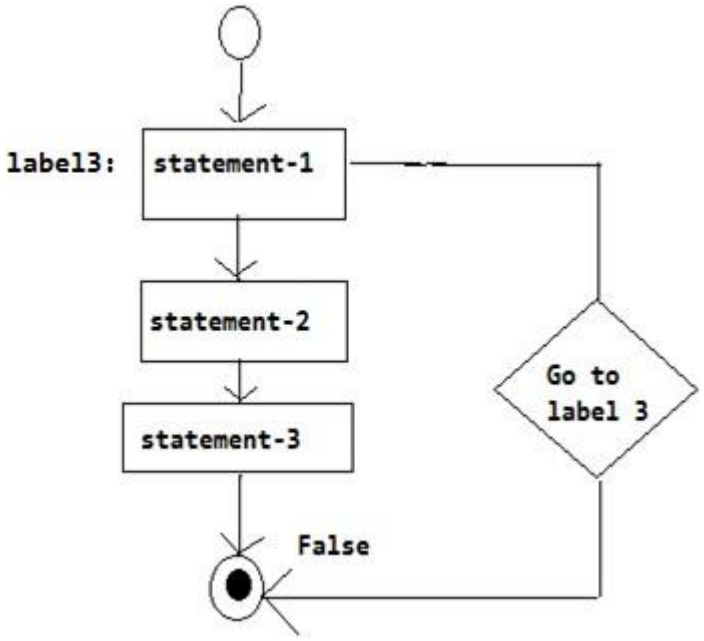
```
goto label;
..
.
label: statement;
```

Or

```
label: statement;
...

...

goto label;
```

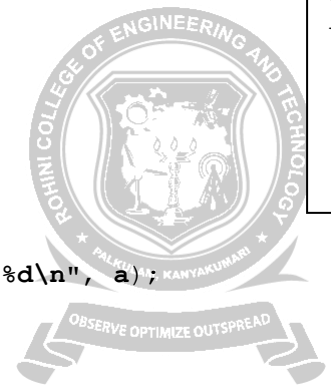


Example program

```
#include <stdio.h>
 int main ()
{
   int a = 10;
     ABCL:do
   {
         if( a == 15)
         {
         a = a + 1;
         goto ABCL;
         }
        printf("value of a: %d\n", a);
        a++;
   }while( a < 20 );
    return 0;
}
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```



program to print 'n' natural number

```
#include<stdio.h>
void main( )
{
     int n,i=1;
     clrscr();
     printf("enter number");
     scanf("%d\t",n);
     printf("natural numbers from 1 to %d", n);
     lb: printf("%d\t",i);
         i++;
     if(i<=n)
         goto lb;
     getch();
 }
```

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD