ONLINE SUPPROT SERVICES

ignou THE PEOPLE'S UNIVERSITY

CERTIFICATE IN INFORMATION TECHNOLOGY

NCERT व्यक्ति निर्माण–राष्ट्र निर्माण

NIRMAN CAMPUS OF EDUCATION RESEARCH & TRAINING

Run & Managed by NASO

IGNOU SC-2281

Jakhepal-Ghasiwala Road, Sunam

For more information visit us at: nirmancampus.co.in

Call us at: 9815098210, 9256278000

CONTROL STATEMENTS:

Flow of execution of program is sequential by nature. To control this execution flow, we use control statement. So, we can say that those statements which are used to control the execution flow in a program are called *Control Statements*. These control statements can be categorized into three divisions:

1. Conditional / Branching / Decision-Making Control Statements.
2. Looping or Repetitive Control Statements.
3. Unconditional Control Statements.

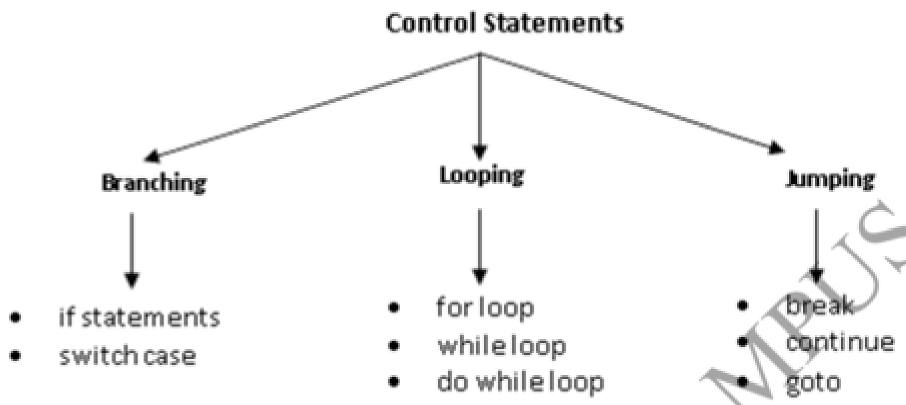


Fig: Classification of Control Statements

BRANCHING CONTROL STATEMENTS

These statements are used for decision making purpose. These are also called ***conditional* or *selection* or *decision – making*** control statements. C language supports the following Branching Control Statements:

i. The 'if else' statements.
ii. The 'switch case' statement.

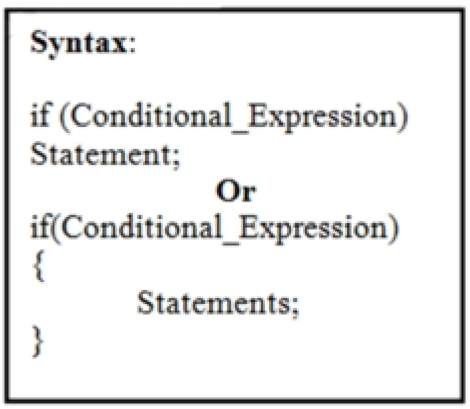
If else statements:

If else statements can be used in the four different ways:

- Only if statement
- If and else statement
- else if ladder statement
- nested if else statement

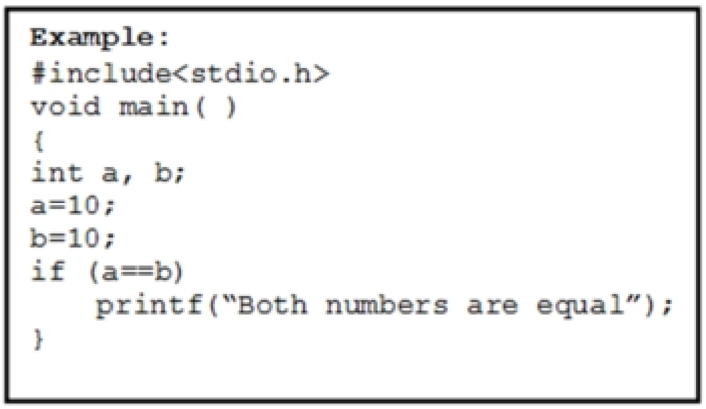
If statement:

It is a branching statement. It is used for decision making purpose. If the given condition is true, it will run the statement otherwise do nothing.
Consider the following example and its syntax:



Syntax:

```
if (Conditional_Expression)
Statement;
                    Or
if(Conditional_Expression)
{
        Statements;
}
```



```
Example:
#include<stdio.h>
void main( )
{
int a, b;
a=10;
b=10;
if (a==b)
    printf("Both numbers are equal");
}
```

If and else statement:

It is a branching statement. It is used for decision making purpose. If the given condition is true it will run the statements1 otherwise it will run the statements2, as shown below in the example:

```
Syntax:

if (Conditional_Expression)
{
        Statements1;
}
else
{
        Statements2;
}
```

```
Example:
void main( )
{
int a, b;
a=10;
b=20;
if (a==b)
      printf("Both numbers are equal");
else
      printf("Numbers are not equal");
}
```

Else if statement:

It is another form of if statement. It is also used for decision making purpose. It is used when we have to test multiple conditions. It is a chain of many if else statements. It executes the first true condition. If no condition is true, it will execute the last else statement. Consider the following syntax and example:

```
Syntax:
if(Conditional_Expression1)
        Statement1;
else if ((Conditional_Expression2)
      Statement2;
    else if ----
    -
        -
        -
          else if ((Conditional_Expression_n)
            Statement n;
```

```
Example:
void main()
{
float per;
per=65;
if(per>=75)
printf("Grade - A");
else if(per>=60)
printf("Grade - B");
else if(per>=40)
      printf("Grade - C");
            else
            printf("Grade - D");
}
```

Nested if else statement:

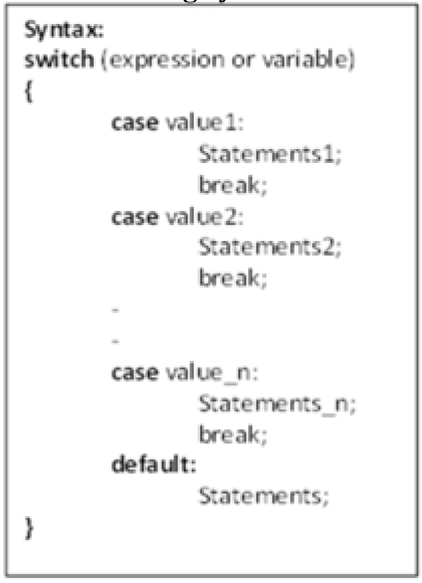
It is another form of branching statement. It is also used for decision making purpose. It is used to test multiple conditions. When one 'if else' statement is used within another if-else statement, it is called Nested if else statement. Consider the following syntax and example:

```
Syntax:
      if(Conditional_Expression)
      {
            if(Conditonal_Expression1)
                  Statement;
            else
                  Statements;
      }
      else
      {
            if(Conditonal_Expression2)
                  Statements;
            else
                  Statements;
      }
```

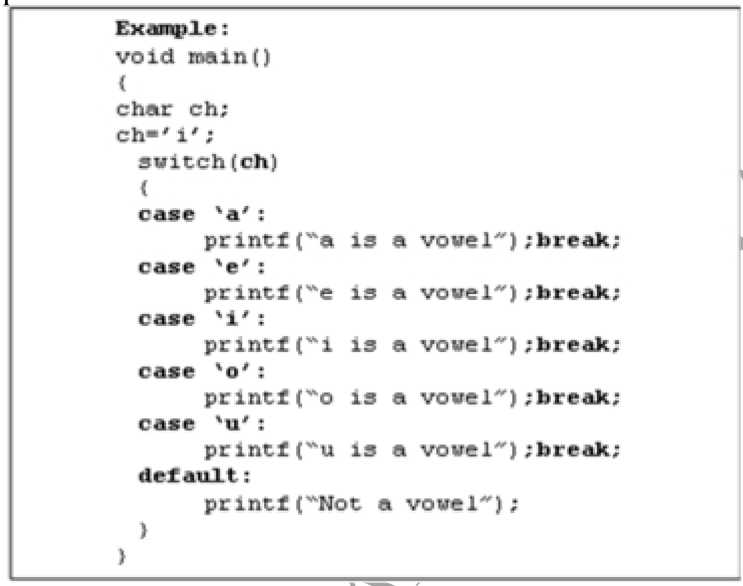
```
Example:
void main()
{
int a=5, b=8, c=6;
      if(a>b)
      {
            if(a>c)
            printf("A is largest");
            else
            printf("C is largest");
      }
      else
      {
            if(b>c)
            printf("B is largest");
            else
            printf("C is largest");
      }
}
```

Switch case:

It is another branching statement. It is also used for the decision making purpose. It is like else if statement. It is used to test multiple conditions. It is used when we have to test a variable or expression against a limited set of constant values. It is used to test integer type operands only. Consider the following syntax and example:



```
Syntax:
switch (expression or variable)
{
        case value1:
                Statements1;
                break;
        case value2:
                Statements2;
                break;
        -
        -
        case value_n:
                Statements_n;
                break;
        default:
                Statements;
}
```



```
Example:
void main()
{
char ch;
ch='i';
  switch(ch)
  {
  case 'a':
       printf("a is a vowel");break;
  case 'e':
       printf("e is a vowel");break;
  case 'i':
       printf("i is a vowel");break;
  case 'o':
       printf("o is a vowel");break;
  case 'u':
       printf("u is a vowel");break;
  default:
       printf("Not a vowel");
  }
}
```

In switch case, expression or variable is matched with each case value. When a match is found, the corresponding block of case is executed. If not case matches with the variable's value, default statement will be executed.

LOOPING STATEMENTS:

These statements are also called **iterative statements**. These are used for repetitions. C provides three types of iterative or looping control structures:

a. The 'for' loop.
b. The 'while' loop.
c. The 'do while' loop.

Any looping control statement, in general, would consist of the following components:

- **Initialization** – It is the starting value of counter variable
- **Test condition** – it is the end value of the counter variable.
- **Iteration** – Increases or decreases the value of counter variable.
- **Body of loop** – These are statements to be executed repeatedly.

Looping control structures may be classified as -

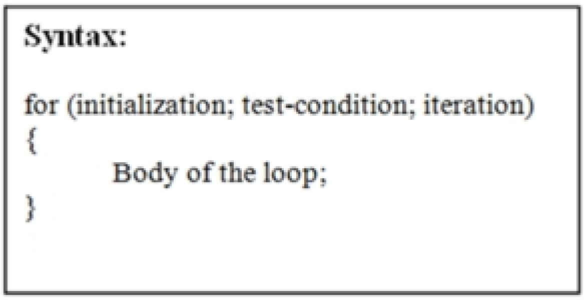
A. Entry - Controlled loop (Pre-Test Loop), and
B. Exit - Controlled loop (Post-Test Loop).

ENTRY CONTROLLED LOOPS:

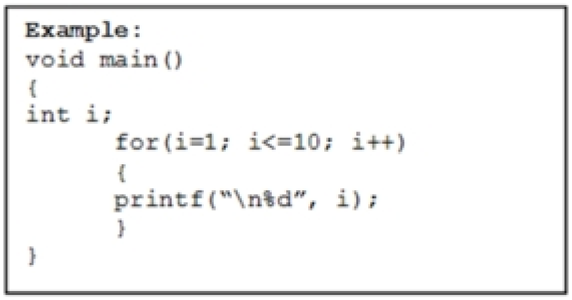
In the entry-controlled loop, the control conditions are tested before the body of loop. The 'for' and 'while' loops are the entry-controlled loops

For loop:

It is a looping control statement. It is used for repeating statements. It is entry controlled loop in which control conditions are tested before the body of loop.



```
Syntax:

for (initialization; test-condition; iteration)
{
        Body of the loop;
}
```



```
Example:
void main()
{
int i;
      for(i=1; i<=10; i++)
      {
      printf("\n%d", i);
      }
}
```

While loop:

It is a looping control statement. It is used for repeating statements. It is entry controlled loop. In this loop condition is tested before the statements (body) of loop.

```
Syntax:

Initialization;
while (condition)
{
        Statements;
        Iteration;
}
```

```
Example:
void main()
{
int i;
        i=1;                    //initialization
        while(i<=10)            //test condition
        {
        printf("\n%d",i);       //statement
        i++;                    //iteration
        }
}
```

EXIT CONTROLLED LOOP:

In the exit-controlled loop, the test condition is performed after the body of the loop. Therefore, this loop must execute at least once, even if the condition is false initially. Do while loop is an example of exit controlled loop.

Do while:

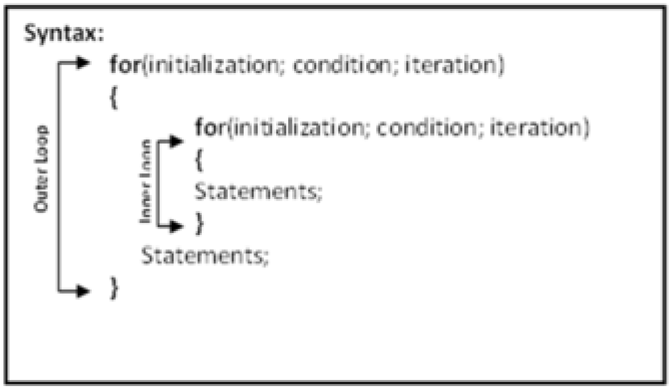
It is a looping control statement. It is also used for repeating statements. It is an exit controlled loop. In this loop, condition is tested after the body of loop. So this loop must execute at least once.

```
Syntax:

Initialization;
do
{
        Statements;
        Iteration;
} while (condition);
```

```
Example:
void main()
{
int i;
        i=11;
        do
        {
                printf("\n%d",i);
                i++;
        }while(i<=10);
}
```

NESTING OF LOOPS

When a loop is used within another loop, It is termed as Nesting of Loops. We can put any loop within another loop. For example, we can put a 'for' loop in another 'for' loop or a 'while' loop etc. Consider the following syntax and example:


Syntax:
for(initialization; condition; iteration)
{
for(initialization; condition; iteration)
{
Statements;
}
Statements;
}
Outer Loop
Inner Loop

```
Example:
void main()
{
int i,j;
  for(i=1;i<=10;i++)
  {
     for(j=1;j<=5;j++)
     {
     printf("\t%d",i*j);
     }
  printf("\n");
  }
}
```

COMPARISON OF WHILE AND DO-WHILE LOOP

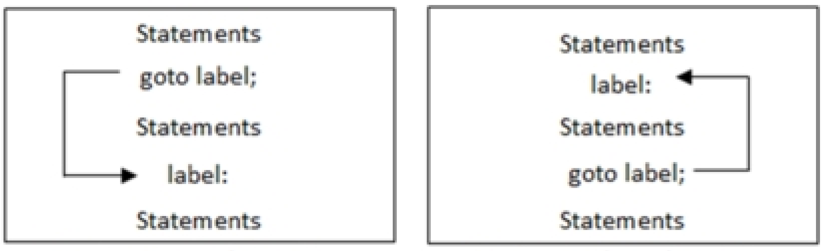
The comparison between 'while' and 'do while' statements is as follow:

1. In 'while' loop, the condition is tested before execution of the body of loop. But, in 'do while' loop, the condition is tested after execution of the body of the loop.
2. The 'do while' loop must execute at least once even if the condition is false initially. But the 'while' loop may not execute if the condition is not satisfied initially.
3. While loop is entry controlled loop and do while loop is exit controlled loop.
4. The 'do while' loop is followed by the semicolon (;) but in the 'while' loop, it is not given.

JUMPING STATEMENTS

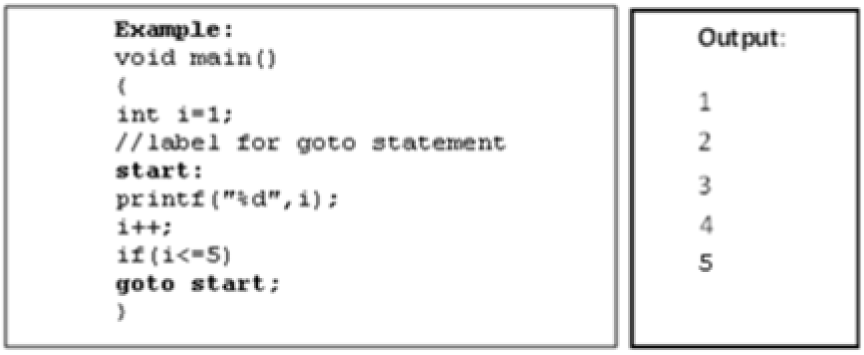
These statements are used to jump the execution control from one to other part of the program. These are also known as **skipping statements**. These statements are goto, continue and break:

The 'goto' statement: It is a jumping statement. It transfers the control at the defined location. Location is defined by the label. It can use either the forward jump or backward jump, as shown in the figure.



Syntax: Forward Jumping Syntax: Backward Jumping

Consider the following example:

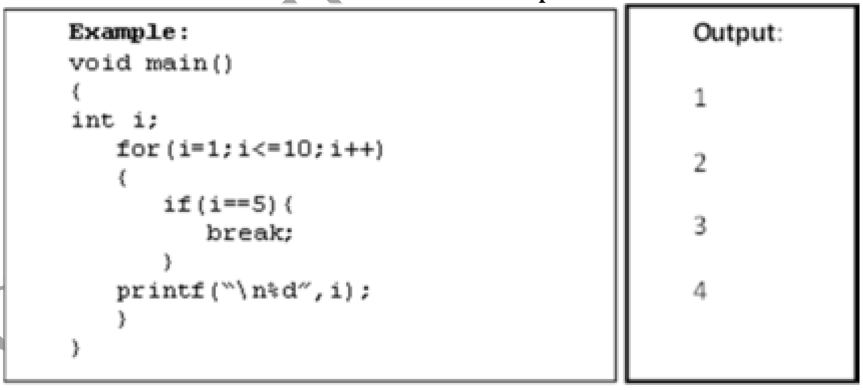


```
Example:
void main()
{
int i=1;
//label for goto statement
start:
printf("%d",i);
i++;
if(i<=5)
goto start;
}
```

```
Output:

1
2
3
4
5
```

The 'break' statement: It is a jumping statement. It takes the control out of the control statement in which it is used. It is widely used in switch statement.

Following example shows the use of break statement in a loop.

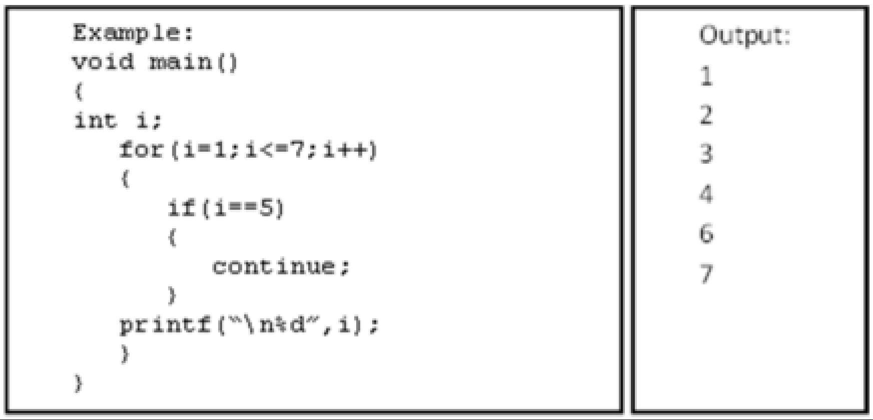


```
Example:
void main()
{
int i;
   for(i=1;i<=10;i++)
   {
      if(i==5){
         break;
      }
   printf("\n%d",i);
   }
}
```

```
Output:

1

2

3

4
```

The 'continue' statement: It is a jumping statement. It takes the control to the next iteration in the loop. It is used in loops.

Following program shows how to use 'continue' statement



```
Example:
void main()
{
int i;
   for(i=1;i<=7;i++)
   {
      if(i==5)
      {
         continue;
      }
   printf("\n%d",i);
   }
}
```

```
Output:
1
2
3
4
6
7
```