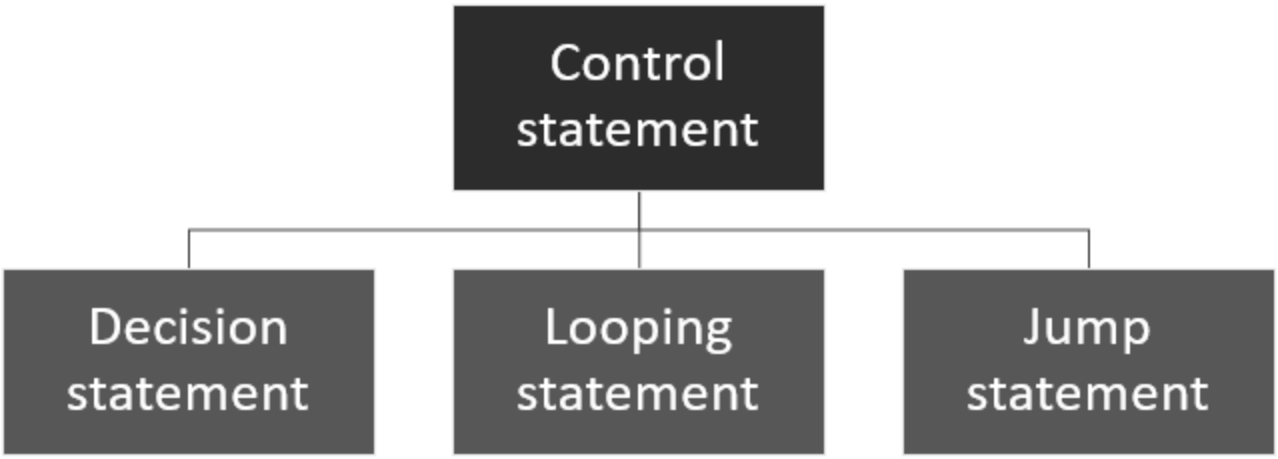
FLOW CONTROL STATEMENTS

Control statements are heart of any programming language. In your programming life you will find 60-70% of your program consist control statements. To master the art of programming, you must have a good control over your program. Control statement is divided in three categories.



Decision statements are also known as conditional or branching statement. In decision statement we will learn about -

- if statement
- if-else-if statement
- Nested if-else statement
- Switch-case

if-else Statement:

The simplest form of the branching statements.

It takes an expression in parenthesis and a statement or block of statements. if the expression is true then the statements gets executed otherwise these statements are skipped and it executes else part.

Syntax:

```
if (condition)     . . . ONLY IF STATEMENT
{
   statements
}
                                  OR
if (condition)     . . . IF ELSE STATEMENT
{
  statements
}

  else
{
  statements
}
```

The following program checks whether the entered number is positive or negative.

```
#include<stdio.h>
int main()

{
    int a;

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

    if(a>0)
     {
      printf( "n The number %d is positive.",a);
     }

    else
     {
      printf("n The number %d is negative.",a);
     }

  return 0;
}
```

If-Else-if statement:

The else-if statement is useful when you need to check multiple conditions within the program, nesting of if-else blocks can be avoided using else-if statement.

Syntax:

```
if (condition1)
{
  statements      //These statements would execute if the condition1 is true
}
else if(condition2)
{
   statements     //These statements would execute if the condition2 is true
}
else if (condition3)
{
   statements     //These statements would execute if the condition3 is true
}
              .
              .
else
{
   statements //These statements would execute if all the conditions return false.
}
```

EXAMPLE:

```c
#include <stdio.h>
int main()
{
   int var1, var2;
   printf("Input the value of var1:");
   scanf("%d", &var1);
   printf("Input the value of var2:");
   scanf("%d",&var2);
   if (var1 !=var2)
   {
        printf("var1 is not equal to var2\n");
   }
   else if (var1 > var2)
   {
        printf("var1 is greater than var2\n");
   }
   else if (var2 > var1)
   {
        printf("var2 is greater than var1\n");
   }
   else
   {
        printf("var1 is equal to var2\n");
   }
   return 0;
}
```

Output:

```
Input the value of var1:12
Input the value of var2:21
var1 is not equal to var2
```

Nested If-else statement:

When an if else statement is present inside the body of another “if” or “else” then this is called nested if else.

Syntax:

```c
if(condition) {
    if(condition2) {
       statements      //Statements inside the body of nested "if"
    }
    else {
     statements        //Statements inside the body of nested "else"
    }
}
else {
    statements         //Statements inside the body of "else"
}
```

Example:

```c
#include <stdio.h>
int main()
{
   int var1, var2;
   printf("Input the value of var1:");
   scanf("%d", &var1);
   printf("Input the value of var2:");
   scanf("%d",&var2);
   if (var1 != var2)
   {
        printf("var1 is not equal to var2\n");
        //Nested if else
        if (var1 > var2)
        {
               printf("var1 is greater than var2\n");
        }
        else
        {
               printf("var2 is greater than var1\n");
        }
   }
   else
   {
        printf("var1 is equal to var2\n");
   }
   return 0;
}
```

Output:

```
Input the value of var1:12
Input the value of var2:21
var1 is not equal to var2
var2 is greater than var1
```

Looping statements

In real life we come across situations when we need to perform a set of task repeatedly till some condition is met. Such as – sending email to all employees, deleting all files, printing 1000 pages of a document.

1. for loop
2. while loop
3. do...while loop

For Loop

For loop is an **entry controlled** looping statement. It is used to repeat set of statements until some condition is met.

Syntax:

```
for (initial value; condition; incrementation or decrementation )
{
  statements;
}
```

Example: for loop to print 1-5 numbers

```
#include<stdio.h>
int main()
{
        int number;
        for(number=1;number<=5;number++)
        {
                printf("%d\n",number);
        }
        return 0;
}
```

Output:

```
1
2
3
4
5
```

While Loop

- A while loop is the most straightforward looping structure.
- A while statement is like a repeating if statement.
- It works Similar as for loop.
- The difference is that after the statements have been executed, the test condition is checked again.
- If it is still true the statements get executed again.
- This cycle repeats until the test condition evaluates to false.

Syntax:

```
while (condition) {
             statements;
}
```

Example: while loop to print 1-5 numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int num=1;
        while(num<=5)
        {
                printf("%d\n",num);
                num++;          //…incrementing operation
        }
        return 0;
}
```

Output:

```
1
2
3
4
5
```

Do-While loop

A do-while loop is similar to the while loop except that the condition is always executed after the body of a loop.

It is also called an exit-controlled loop.

The body of do-while loop is executed at least once.

Syntax:

```
do {
  statements
} while (expression);
```

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int num=1;
        do
        {
                printf("%d\n",2*num);
                num++;
        }while(num<=10);
```

```
        return 0;
}
```

Output:

```
2
4
6
8
10
12
14
16
18
20
```

Switch Statement:

- The switch statement allows us to execute one code block among many alternatives.
- A switch statement allows a variable to be tested for equality against a list of values.
- Each value is called a case, and the variable being switched on is checked for each switch case.

Syntax:

```
switch (i)
{
    case 1:           // code to be executed if i = 1;
       statement
       break;
    case 2:           // code to be executed if i = 2;
       statement
       break;
              .
              .
              .
    case n:            // code to be executed if i = n;
       statement
       break;
    default:           // code to be executed if i doesn't match any cases
}
```

Example:

```
#include <stdio.h>
int main()
{
   int x = 2;
   switch (x)
   {
       case 1: printf("Choice is 1");
               break;
       case 2: printf("Choice is 2");
                break;
       case 3: printf("Choice is 3");
               break;
       default: printf("Choice other than 1, 2 and 3");
                break;
   }
   return 0;
}
```

Output:

Choice is 2

goto statement

The goto statement is known as jump statement in C.

The goto statement allows us to transfer control of the program to the specified label

Syntax:

```
goto label;

..

.

label: statement;
```

Example:

```
#include <stdio.h>
int main()
{
   int sum=0;
   for(int i = 0; i<=10; i++){
         sum = sum+i;
         if(i==5){
            goto addition;
         }
   }

   addition:
   printf("%d", sum);

   return 0;
}
```