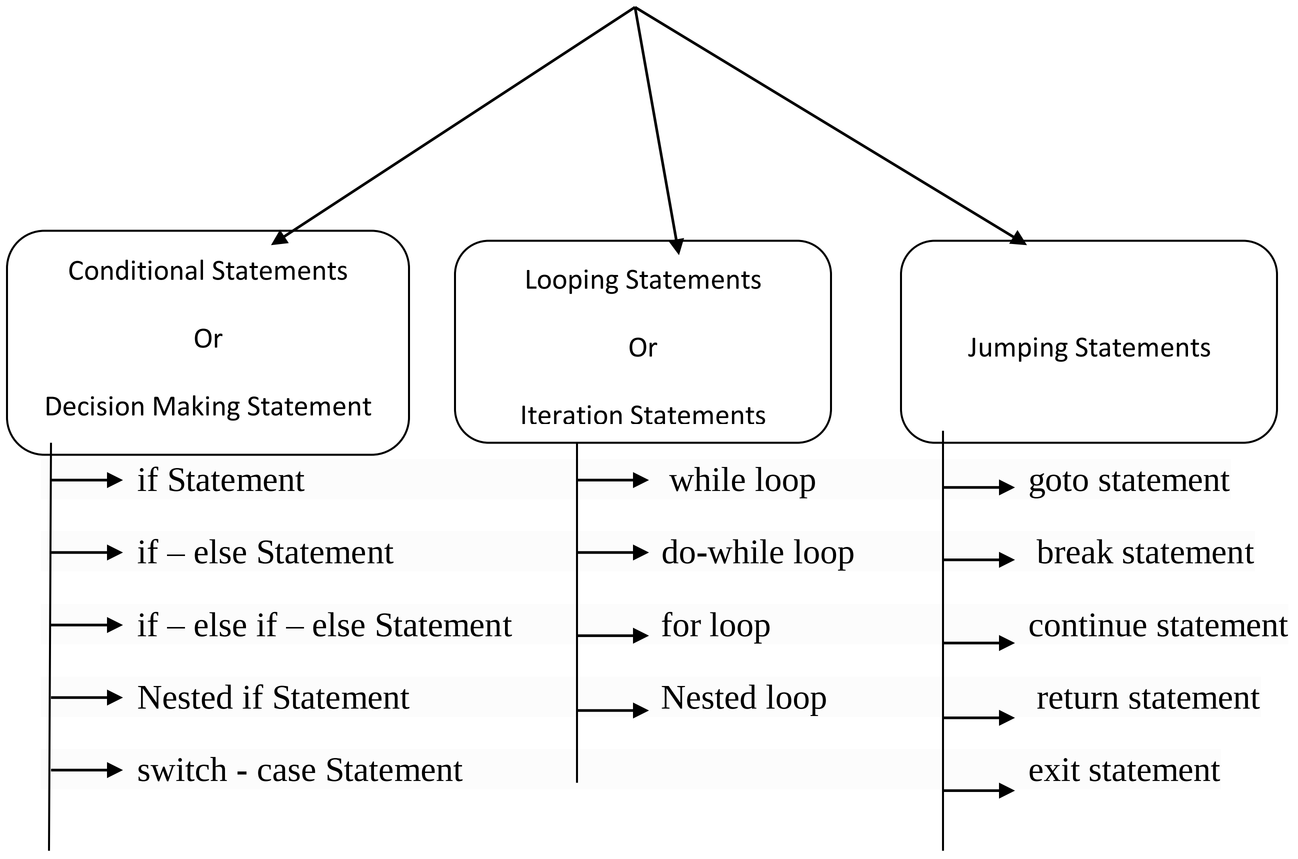
Unit – 3

Control Statements

Every statement in a computer is executed based on pre-defined rules. The control flow is also based on logic. You find a necessity to execute a few customized logics. The control enters the statements block and gets executed if the logic is satisfied. Hence, they are called control statements. In simple words, Control statements in C help the computer execute a certain logical statement and decide whether to enable the control of the flow through a certain set of statements or not.



Conditional Statements

Conditional statements help you to make a decision based on certain conditions. These conditions are specified by a set of conditional statements having Boolean expressions which are evaluated to a Boolean value true or false. There are following types of conditional statements in C.

If Statement

The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition. It is mostly used in the scenario where we need to perform the different operations for the different conditions. The syntax of the if statement is given below.

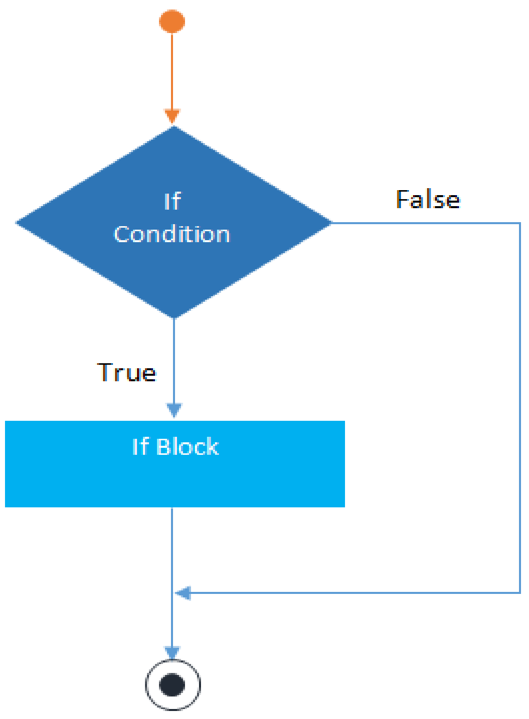
Syntax 1:

```
if(condition)
Statement;
```

Syntax 2:

```
if (condition)
{
Statement 1;
}
Statement x;
```

Flow chart of if statement:



If statement

Example 1:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        clrscr();
        printf(“Enter a character :”);
        scanf(“%c”, &ch);
        if(ch==’a’)
        printf(“Amit Kumar”);
        getch();
}
```

Output:

Run 1:

```
Enter a character : a
Amit Kumar
```

Run 2:

```
Enter a character : r
```

Example 2:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number :");
        scanf("%d", &n);
        if(n>100)
        printf("Number is greater than 100");
        getch();
}
```

Output:

Run 1:

```
Enter a number : 140
Number is greater than 100
```

Run 2:

```
Enter a number : 40
```

Example 3:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number :");
        scanf("%d", &n);
        if(n<100)
        printf("Number is smaller than 100");
        getch();
}
```

Output:

Run 1:

```
Enter a number : 50
Number is smaller than 100
```

Run 2:

```
Enter a number : 150
```

Example 4:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n1, n2;
        clrscr();
        printf("Enter two numbers :");
        scanf("%d%d", &n1, &n2);
        if(n1<n2)
        printf("%d is less than %d", n1, n2);
        getch();
}
```

Output:

Run 1:

```
Enter two numbers  : 140          150
140 is less than 150
```

Run 2:

```
Enter two numbers  : 240          50
```

Example 5:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n1, n2;
        clrscr();
        printf("Enter two numbers :");
        scanf("%d%d", &n1, &n2);
        if(n1>n2)
        printf("%d is greater than %d", n1, n2);
        getch();
}
```

Output:

Run 1:

```
Enter two numbers  : 160        150
160 is greater than 150
```

Run 2:

```
Enter two numbers  : 20         50
```

If-else Statement

The if-else statement is used to perform two operations for a single condition. The if-else statement is an extension to the if statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition.

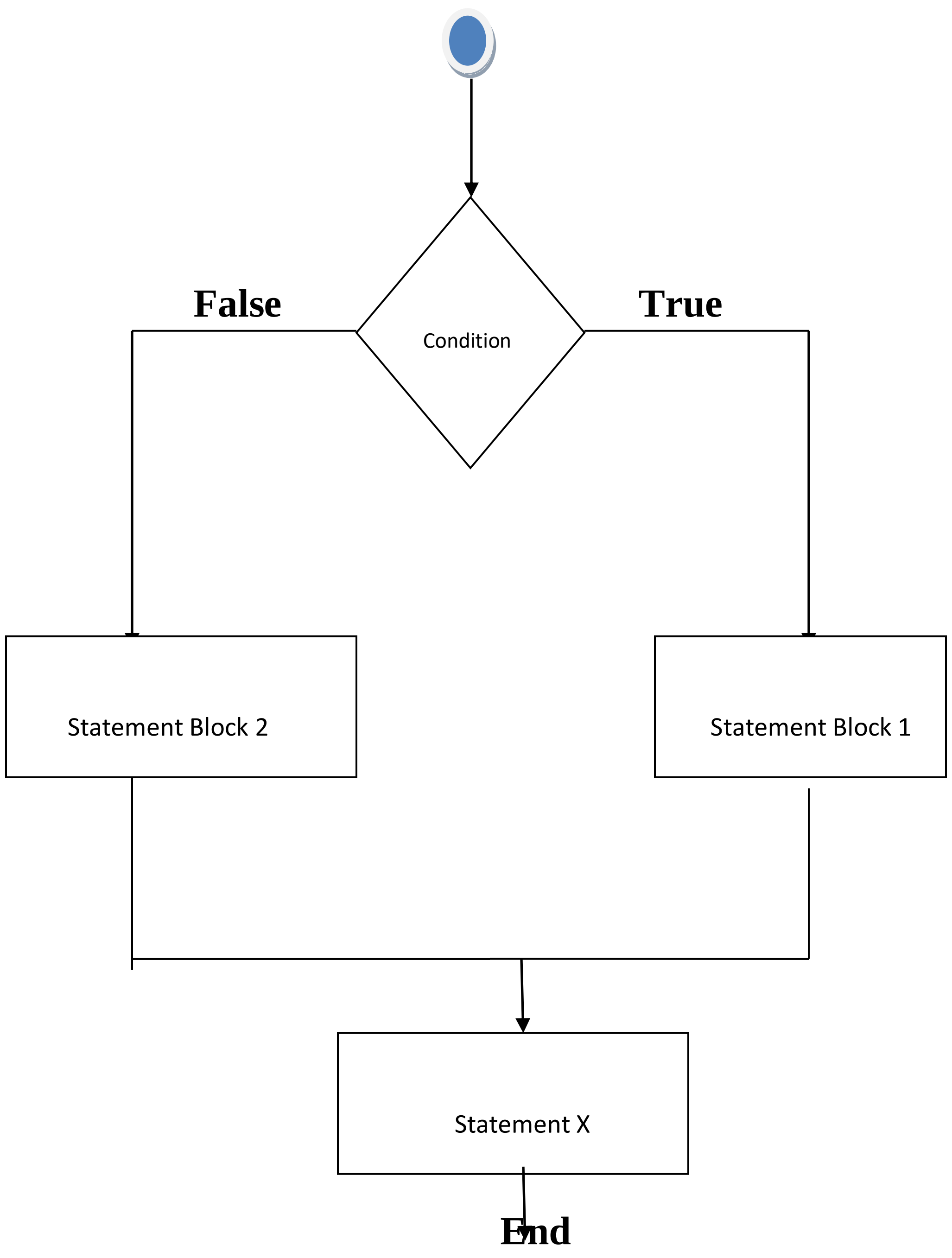
Syntax 1

```
if(condition)
        statement 1;
else
        statement 2;
statement x;
```

Syntax 2

```
if(condition)
{
----------- ⎫
----------- ⎬    Statement Block 1
----------- ⎭
}
else
{
----------⎫
----------⎬              Statement Block 2
----------⎭
}
Statement x;
```

Flow chart of if – else statement



Example 1:

WAP to check the given number is greater or smaller than 100.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number =");
        scanf("%d", &n);
        if(n>100)
        printf("Number is greater than 100");
        else
        printf("Number is smaller than 100");
        getch();
}
```

Output:

Run 1:

```
Enter a number = 120
Number is greater than 100
```

Run 2:

```
Enter a number = 20
Number is smaller than 100
```

Example 2:

WAP to check the greater number between has given two numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n1, n2;
        clrscr();
        printf("Enter two numbers =");
        scanf("%d%d", &n1, &n2);
        if(n1>n2)
        printf("n1 is greater");
        else
        printf("n2 is greater");
        getch();
}
```

Output:

Run 1:

```
Enter two numbers = 120     220
n2 is greater
```

Run 2:

```
Enter two numbers = 220     120
n1 is greater
```

Example 3:
WAP to check whether given number is Even or Odd

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number =");
        scanf("%d", &n);
        if(n%2==0)
        printf("Number is Even");
        else
        printf("Number is Odd");
        getch();
}
```

Output:

Run 1:

```
Enter a number = 20
Number is Even
```

Run 2:

```
Enter a number = 21
Number is Odd
```

Example 4:
WAP to check whether given year is Leap or not leap

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf(“Enter a year =”);
        scanf(“%d”, &n);
        if(n%4==0)
        printf(“Leap Year”);
        else
        printf(“Not a Leap Year”);
        getch();
}
```

Output:

Run 1:

```
Enter a year = 2020
Leap Year
```

Run 2:

```
Enter a year = 2021
Not a Leap Year
```

Example 5:
WAP to check whether given number is +ve or -ve.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number =");
        scanf("%d", &n);
        if(n>0)
        printf("Positive Number");
        else
        printf("Negative Number");
        getch();
}
```

Output:

Run 1:

```
Enter a number = 20
Positive Number
```

Run 2:

```
Enter a year = -21
Negative Number
```

Example-6:

C program to check whether a character is an alphabet or not

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        clrscr();
        printf("Enter a character =");
        scanf("%c", &ch);
        if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z') )
        printf("It is an alphabet");
        else
        printf("It is not an alphabet");
        getch();
}
```

Output:

Run 1:

```
Enter a character = R
It is an alphabet
```

Run 2:

```
Enter a year = 6
It is not an alphabet
```

Example-7:

Program to check whether a person is eligible to vote or not.

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int age;
   clrscr();
   printf("Enter your age =");
   scanf("%d",&age);
   if(age>=18)
   {
      printf("You are eligible to vote");
   }
   else
   {
      printf("Sorry! you can't vote");
   }
getch();
}
```

Output:

Run 1:

```
Enter your age = 20
You are eligible to vote
```

Run 2:

```
Enter your age = 17
Sorry! Yor cant’s vote
```

if – else if – else Statement

The if-else-if ladder statement is an extension to the if-else statement. It is used in the scenario where there are multiple cases to be performed for different conditions. In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed. There are multiple else-if blocks possible.

Syntax 1:

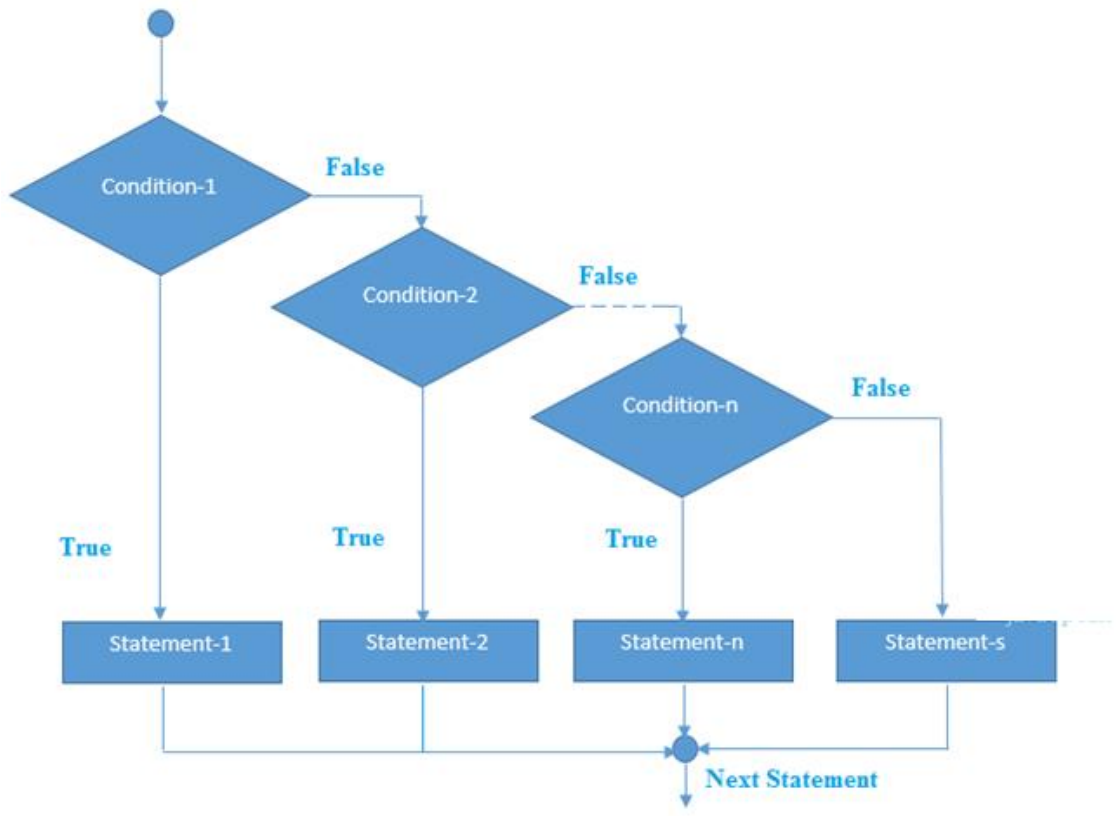
```
if(condition 1)
        Statement 1;
else if(condition 2)
        Statement 2;
else if(condition 3)
        Statement 3;
.
.
.
else if(condition n)
        Statement n;
else
        Statement s;
Statement X;
```

Syntax 2:

```
if(condition 1)
{
-----------
-----------
}
else if(condition 2)
{
-----------
-----------
}
else if(condition 3)
{
```

```
------------
------------
}
.
.
.
else if(condition n)
{
------------
------------
}
else
{
------------
------------
}
Statement X;
```

Flow Chart of if – else if – else statement



Example 1:

WAP to check given number is +ve, -ve or zero.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number =");
        scanf("%d", &n);
        if(n>0)
        printf("Positive Number");
        else if(n<0)
        printf("Negative Number");
        else
        printf("Number is zero");
        getch();
}
```

Output:

Run 1:

```
Enter a number: 10
Positive Number
```

Run 2:

```
Enter a number: -10
Negative Number
```

Run 3:

```
Enter a number: 0
Number is zero
```

Example 2:

WAP to find out largest number among three numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n1, n2, n3;
        clrscr();
        printf("Enter any three number =");
        scanf("%d%d%d", &n1, &n2, &n3);
        if(n1>n2 && n1>n3)
        printf("%d is greater", n1);
        else if(n2>n1 && n2>n3)
        printf("%d is greater", n2);
        else
        printf("%d is greater", n3);
        getch();
}
```

Output:

Run 1:

```
Enter any three number = 30 20 25

30 is greater
```

Run 2:

```
Enter any three number = 30 40 25

40 is greater
```

Run 3:

```
Enter any three number = 30 20 45

45 is greater
```

Example 3:

WAP to check given number is one digit, two digit, three digit and four digit number

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number =");
        scanf("%d", &n);
        if(n>=0 && n<=9)
        printf("One digit number");
        else if(n>=10 && n<=99)
        printf("Two digit number");
        else if(n>=100 && n<=999)
        printf("Three digit number");
        else if(n>=1000 && n<=9999)
        printf("Four digit number");
        else
        printf("Invalid Number");
        getch();
}
```

Output:

Run 1:

```
Enter a number = 6

One digit number
```

Run 2:

```
Enter a number = 16

Two digit number
```

Run 3:

```
Enter a number = 612

Three digit number
```

Run 4:

```
Enter a number = 6123

Four digit number
```

Run 5:

```
Enter a number = 12612

Invalid Number
```

Example 4:

WAP to print week day name according to the given weekday number

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int day;
	clrscr();
	printf(“Enter the weekday number =”);
	scanf(“%d”, &day);
	if(day==1)
	printf(“Sunday”);
	else if(day==2)
	printf(“Monday”);
	else if(day==3)
	printf(“Tuesday”);
	else if(day==4)
	printf(“Wednesday”);
	else if(day==5)
```

```
        printf(“Thursday”);
        else if(day==6)
        printf(“Friday”);
        else if(day==7)
        printf(“Saturday”);
        else
        printf(“Invalid Week day Number”);
        getch();
}
```

Output:

Run 1:

```
Enter the weekday number : 1

Sunday
```

Run 2:

```
Enter the weekday number : 2

Monday
```

Run 3:

```
Enter the weekday number : 3

Tuesday
```

Run 4:

```
Enter the weekday number : 4

Wednesday
```

Run 5:

```
Enter the weekday number : 5

Thursday
```

Run 6:

```
Enter the weekday number : 6

Friday
```

Run 7:

```
Enter the weekday number : 7

Saturday
```

Run 8:

```
Enter the weekday number : 8

Invalid Weekday number
```

Example 4:

WAP to print month name according to the given month number

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int month;
        clrscr();
        printf("Enter the month number =");
        scanf("%d", &month);
        if(month==1)
        printf("January");
        else if(month==2)
        printf("February");
        else if(month==3)
        printf("March");
        else if(month==4)
        printf("April");
        else if(month==5)
        printf(“May”);
        else if(month==6)
        printf(“June”);
        else if(month==7)
        printf(“July”);
        else if(month==8)
        printf(“August”);
        else if(month==9)
        printf(“September”);
        else if(month==10)
        printf(“October”);
        else if(month==11)
        printf(“November”);
        else if(month==12)
        printf(“December”);
        else
        printf(“Invalid month Number”);
        getch();
}
```

Output:

Run 1:

```
Enter the month number: 1
January
```

Run 2:

```
Enter the month number: 2
February
```

Run 3:

```
Enter the month number: 3
March
```

Run 4:

```
Enter the month number: 4
April
```

Run 5:

```
Enter the month number: 5
May
```

Run 6:

```
Enter the month number: 6
June
```

Run 7:

```
Enter the month number: 7
July
```

Run 8:

```
Enter the month number: 8
August
```

Run 9:

```
Enter the month number: 9
September
```

Run 10:

```
Enter the month number: 10
October
```

Run 11:

```
Enter the month number: 11
```

November

Run 12:

```
Enter the month number: 12

December
```

Run 13:

```
Enter the month number: 13

Invalid month number
```

Exercise for students:

1. WAP to find out smallest number among three numbers.
2. WAP to check given character is vowel or consonant.
3. WAP to check given alphabet is lower case, upper case, digit or special alphabet.

C Switch Statement

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable. Here, We can define various statements in the multiple cases for the different values of a single variable.

```
switch(expression)
{
case value1:
 //code to be executed;
 break;
case value2:
 //code to be executed;
 break;
......
.
.
.
case value n:
 //code to be executed;
 break;
default:
 code to be executed if all cases are not matched;
}
```

Rules for switch statement in C language

1) The *switch expression* must be of an integer or character type.

2) The *case value* must be an integer or character constant.

3) The *case value* can be used only inside the switch statement.

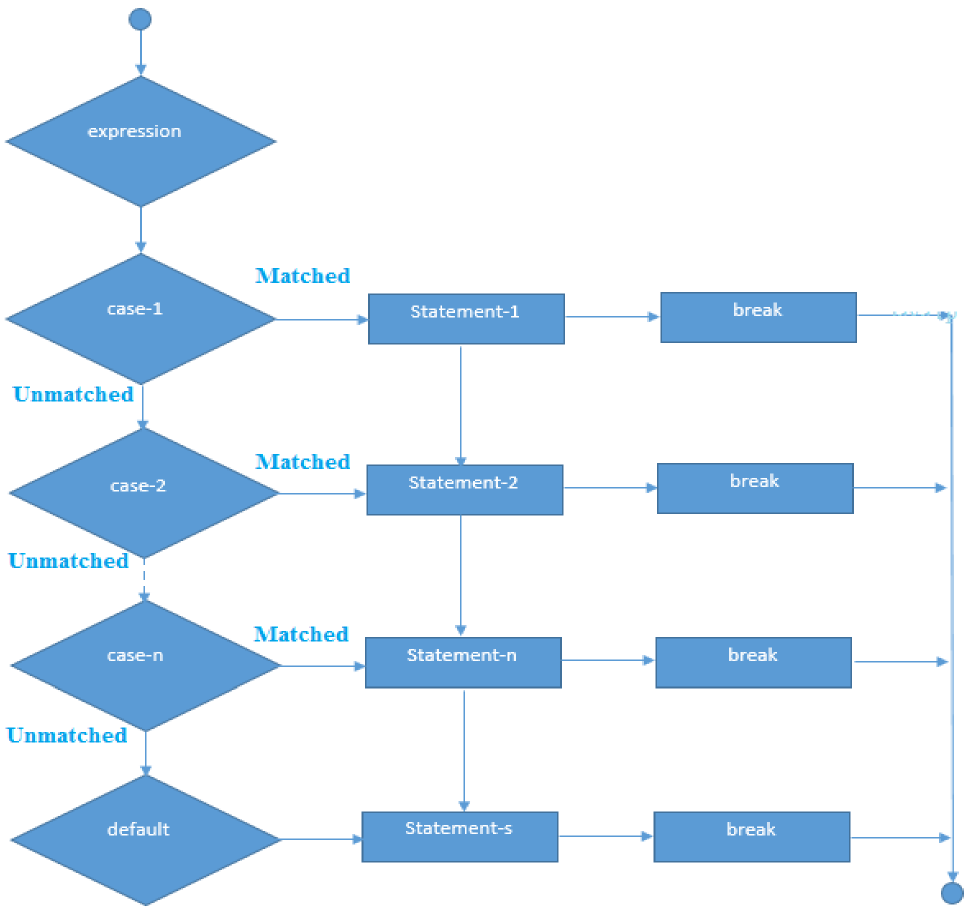
4) The *break statement* in switch case is not must. It is optional. If there is no break statement found in the case, all the cases will be executed present after the matched case.

We are assuming that there are following variables.

int x,y,z;
char a,b;
float f;

Valid Switch	Invalid Switch	Valid Case	Invalid Case
switch(x)	switch(f)	case 3;	case 2.5;
switch(x>y)	switch(x+2.5)	case 'a';	case x;
switch(a+b-2)		case 1+2;	case x+2;
switch(func(x,y))		case 'x'>'y';	case 1,2,3;

Flow Chart of Switch Case:



Example 1:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int number;
printf("Enter a number:");
scanf("%d", &number);
switch(number)
{
case 10:
printf("number is equals to 10");
break;
case 50:
printf("number is equal to 50");
break;
case 100:
printf("number is equal to 100");
break;
default:
printf("number is not equal to 10, 50 or 100");
}
getch();
}
```

Output

Run 1:

```
Enter a number:4
number is not equal to 10, 50 or 100
```

Run 2:

```
Enter a number:50
number is equal to 50
```

Example 2:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int day;
clrscr();
printf("Enter the week day number =");
scanf("%d", &day);
switch(number)
{
case 1:
printf("Sunday");
break;
case 2:
printf("Monday");
break;
case 3:
printf("Tuesday");
break;
case 4:
printf("Wednesday");
break;
case 5:
printf("Thursday");
break;
case 6:
printf("Friday");
break;
case 7:
printf("Saturday");
break;
default:
printf("Invalid Weekday number");
}
getch();
}
```

Example 3:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch;
clrscr();
printf("Enter a character =");
scanf("%c", &ch);
switch(ch)
{
case ‘a’:
case ‘A’
printf("Vowel");
break;
case ‘e’:
case ‘E’:
printf("Vowel");
break;
case ‘i’:
case ‘I’
printf("Vowel");
break;
case ‘o’:
case ‘O’:
printf("Vowel");
break;
case ‘u’:
case ‘U’:
printf("Vowel");
break;
default:
printf("Consonants");
}
getch();
}
```

Example 4:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char op;
clrscr();
printf("Enter an arithmetic operator (+, -, *, /, %) =");
scanf("%c", &op);
printf(“Enter two numbers =”);
scanf(“%d%d”, &a, &b);
switch(op)
{
case ‘+’:
c=a+b;
printf(“\n The sum is = %d”, c);
break;
case ‘-’:
c=a-b;
printf(“\n The sub is = %d”, c);
break;
case ‘*’:
c=a*b;
printf(“\n The mul is = %d”, c);
break;
case ‘/’:
c=a/b;
printf("\n The div is = %d", c);
break;
case '%':
c=a%b;
printf("\n The mod is = %d", c);
break;

default:
printf("Invalid Operator");
}
getch();
}
```

Nested Loops in C

C supports nesting of loops in C. **Nesting of loops** is the feature in C that allows the looping of statements inside another loop. Any number of loops can be defined inside another loop, i.e., there is no restriction for defining any number of loops. The nesting level can be defined at n times. You can define any type of loop inside another loop; for example, you can define '**while**' loop inside a '**for**' loop.

Syntax of Nested loop

```
Outer_loop
{
   Inner_loop
   {
        // inner loop statements.
   }
      // outer loop statements.
}
```

Outer_loop and **Inner_loop** are the valid loops that can be a 'for' loop, 'while' loop or 'do-while' loop.

Syntax 1

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

```
for ( initialization; condition; increment/decrement )
```

```
{

   for ( initialization; condition; increment/decrement )
   {
      //statements of inner loop;
   }
     // statements of outer loop;
}
```

Syntax 2

The syntax for a **nested while loop** statement in C programming language is as follows −

```
while(condition)
{

   while(condition)
   {
      //statements of inner loop
   }
      //statements of inner loop
}
```

Syntax 3

The syntax for a **nested do...while loop** statement in C programming language is as follows −

```
do
{
   //statements of inner loop
   do
    {
    //statements of inner loop
    }while( condition );

}while( condition );
```

Example of nested while loop

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int i=1, j=2;
   clrscr();
   while(i<=10)
   {
```

```
             while(j<=20)
             {
                 printf("\n%d", j);
                 J=j+2;
             }
    printf("\n%d", i);
    i++;
}
getch();
}
```

Example of nested do-while loop

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int i=1, j=2;
   clrscr();
  do
   {
             printf("\n%d", i);
             i++;
             do
             {
                 printf("\n%d", j);
                 j=j+2;
              } while(j<=20);
             printf("\n");
} while(i<=10);
getch();
}
```

Example of nested for loop

```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
  int i, j;
  clrscr();
  for(i=1;i<=5;i++)
  {
    for(j=1;j<=i; j++)
    {
            printf("  %d ", j);
}
printf("\n");
getch();
}
```

Jumping Statement

Jump Statement makes the control jump to another section of the program unconditionally when encountered. It is usually used to terminate the **loop** or **switch-case** instantly. It is also used to escape the execution of a section of the program.

break statement

The break is a keyword in C which is used to bring the program control out of the loop. The break statement is used inside loops or switch statement. The break statement breaks the loop one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break statement in C can be used in the following two scenarios:

1. With switch case
2. With loop

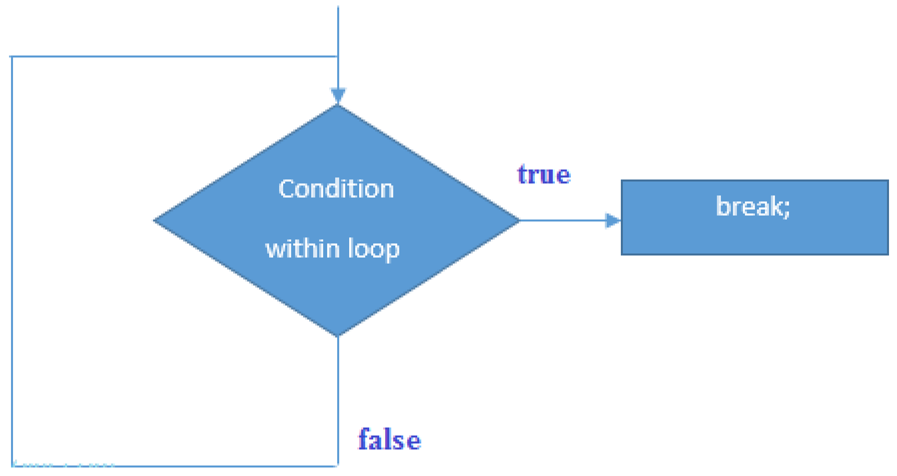


Figure: Flowchart of break statement

Example

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    int i;
    clrscr();
    for(i = 1; i<=10; i++)
    {
        printf("%d ",i);
        if(i == 5)
        break;
    }
    printf("came outside of loop i = %d",i);
    getch();
}
```

Output

```
1 2 3 4 5 came outside of loop i = 5
```

Example

```
#include<stdio.h>
```

```
#include<conio.h>

void main ()
{
   int n=2,i,choice;
   do
   {
       i=1;
       while(i<=10)
       {
           printf("%d X %d = %d\n",n,i,n*i);
           i++;
       }
       printf("do you want to continue with the table of %d , enter any nonzero value to continue.",n+1);
       scanf("%d",&choice);
       if(choice == 0)
       {
           break;
        }
       n++;
   }while(1);
getch();
}
```

Output

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
do you want to continue with the table of 3 , enter any non-zero value to continue.3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
```

```
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
do you want to continue with the table of 4 , enter any non-zero value to continue.0
```

continue statement

The **continue statement** in C language is used to bring the program control to the beginning of the loop. The continue statement skips some lines of code inside the loop and continues with the next iteration. It is mainly used for a condition so that we can skip some code for a particular condition.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();
for(i=1;i<=10;i++)
{
if(i==5)
{
continue;
}
printf("%d \n",i);
}
getch();
}
```

Output

```
1
2
3
4
6
7
8
9
10
```

goto statement

The goto statement is known as jump statement in C. As the name suggests, goto is used to transfer the program control to a predefined label. The goto statment can be used to repeat some part of the code for a particular condition. It can also be used to break the multiple loops which can't be done by using a single break statement.

Syntax:

```
label:
//some part of the code;
goto label;
```

goto example

Let's see a simple example to use goto statement in C language.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int num,i=1;
  clrscr();
  printf("Enter the number whose table you want to print?");
  scanf("%d",&num);
  table:
  printf("%d x %d = %d\n",num,i,num*i);
  i++;
  if(i<=10)
  goto table;
  getch();
}
```

Output:

```
Enter the number whose table you want to print?10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
```

```
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

exit() function in C

The **exit() function** is used to terminate a process or function calling immediately in the program. It means any open file or function belonging to the process is closed immediately as the exit() function occurred in the program. The exit() function is the standard library function of the C, which is defined in the <**stdlib.h> or <process.h>** header file. So, we can say it is the function that forcefully terminates the current program and transfers the control to the operating system to exit the program. The exit(0) function determines the program terminates without any error message, and then the exit(1) function determines the program forcefully terminates the execution process.

The **`exit()`** function has no return type.

Example:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main ()
{
int i, num;
clrscr();
printf ( " Enter the last number: ");
scanf ( " %d", &num);
for ( i = 1; i<num; i++)
{
if ( i == 6 )
{
exit(0);
}
else
{
printf  (" \n Number is %d", i);
}
getch();
}
```

Output

```
Enter the last number: 10

 Number is 1
 Number is 2
 Number is 3
 Number is 4
 Number is 5
```

Looping Statements

The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times. For example, if we need to print the first 10 natural numbers then, instead of using the printf statement 10 times, we can print inside a loop which runs up to 10 iterations.

Advantage of loops in C

1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again.

Types of C Loops

There are three types of loops in C language that is given below:

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

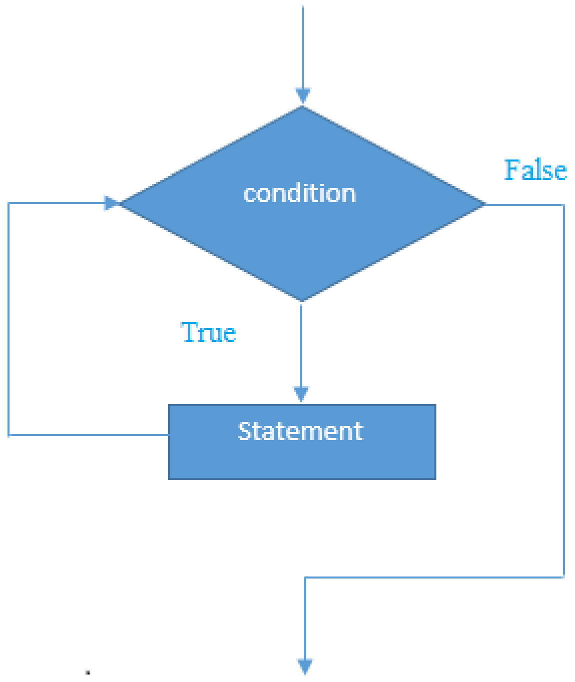
while loop

The while loop in c is to be used in the scenario where we don't know the number of iterations in advance. The block of statements is executed in the while loop until the condition specified in the while loop is satisfied. It is also called a pre-tested loop or entry controlled loop.

Syntax

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

Flowchart of while loop



Statement – X

Example 1:

WAP to print the counting from 1 to 10.

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

Output

```
1
2
3
4
5
6
7
8
9
10
```

Example 2:

WAP to print the counting from 10 to 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n=10;
clrscr();
while(n>=1)
{
printf("\n%d" ,n);
n--;
}
getch();
}
```

Output

```
10
9
8
7
6
5
4
3
2
1
```

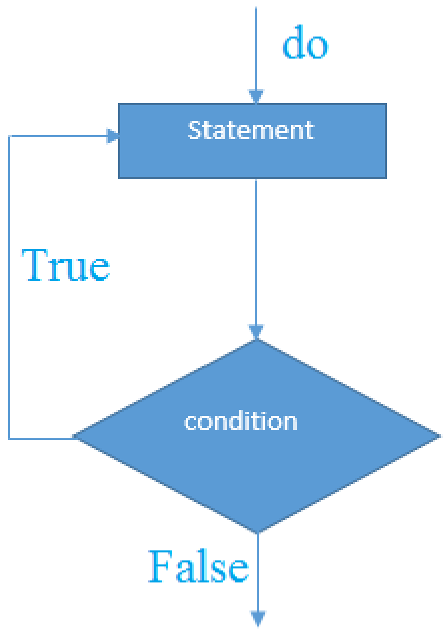
do-while loop in C

The do-while loop continues until a given condition satisfies. It is also called post tested loop. It is used when it is necessary to execute the loop at least once. This loop is also called post tested loop or exit-controlled loop.

Syntax

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

Flowchart of do-while loop



Statement – X

Example 1:

WAP to print the counting from 1 to 10.

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

Output

```
1
2
3
4
5
6
7
8
9
10
```

Example 2:

WAP to print the counting from 10 to 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n=10;
clrscr();
do
{
printf("\n%d" ,n);
n--;
} while(n>=1);
  getch();
}
```

Output

```
10
9
8
7
6
5
4
3
2
1
```

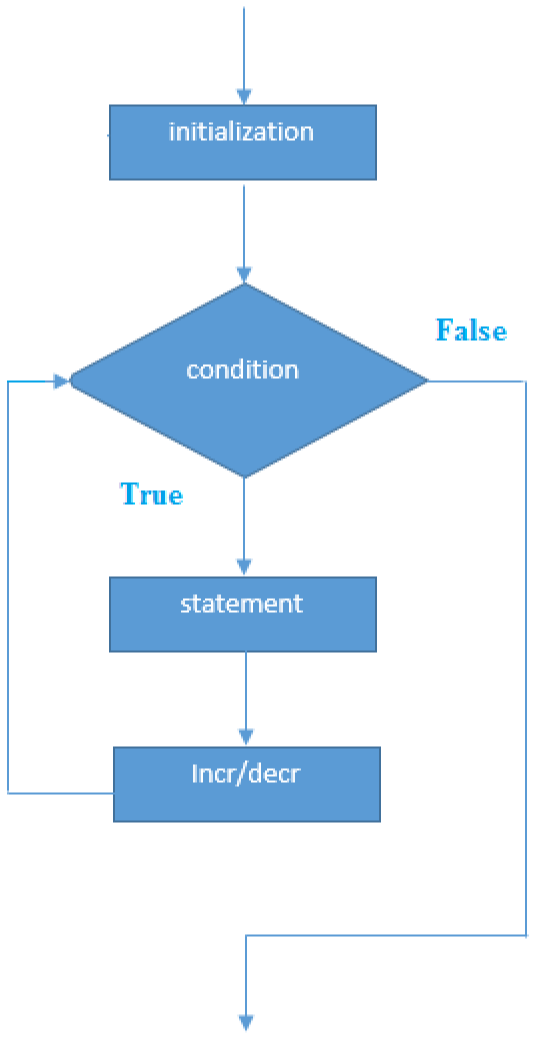
for loop in c

The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. It is better to use for loop if the number of iteration is known in advance.

Syntax

```
for(initialization; condition; incr/decr)
{
//statements
}
```

Flowchart of for loop



Statement – X

Example 1:

WAP to print the counting from 1 to 10.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
clrscr();
for(n=1; n<=10; n++)
{
printf("\n%d" ,n);
}
  getch();
}
```

Output

```
1
2
3
4
5
6
7
8
9
10
```

Example 2:

WAP to print the counting from 10 to 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
clrscr();
for(n=10; n>=1; n--)
{
printf("\n%d" ,n);
}
  getch();
}
```

Output

```
10
9
```

```
8
7
6
5
4
3
2
1
```

Problems related to Looping statements:

1. WAP to print the counting from 1 to 10.
2. WAP to print the counting from 10 to 1.
3. WAP to print the counting from 1 to last number.
4. WAP to print the counting from starting to end number.
5. WAP to print the natural number up to given number.
6. WAP to print the sum of 1 to 10 counting.
7. WAP to print the sum of 1 to last number.
8. WAP to print the sum of starting number to end number.
9. WAP to print the table of any number.
10. WAP to print the all even numbers up to has given number.
11. WAP to print the all odd numbers up to has given number.
12. WAP to print the reverse of given number.
13. WAP to count the digits of a given number.
14. WAP to print the sum of all digits of a given number.
15. WAP to print the sum of first and last digit of a given number.
16. WAP to count even and odd digits from a number.
17. WAP to print the sum of even digits and odd digits of a given number.
18. WAP to calculate the factorial of given number.
19. WAP to print the Fibonacci series.
20. WAP to check whether given number is prime or not.
21. WAP to check whether given number is Armstrong or not.
22. WAP to check whether given number is palindrome or not.
23. WAP to check whether given number is perfect or not.
24. WAP to calculate the power of given number.
25. WAP to print the sine series.
26. WAP to print the cosine series.
27. WAP to print the factors of given number.