

Unit 2

Basic of C

Operator: An operator is a Symbol that operates. An operator acts on some variables called operands to get the desired result. An operator is a symbol that tells the computer to perform mathematical or logical operations on operands. Operators are used in programs to manipulate data. C operators can be classified into several categories:

Types of Operators:

- 1) Arithmetic Operators
- 2) Relational Operators
- 3) Logical Operators
- 4) Assignment Operators
- 5) Unary Operators (increment/decrement)
- 6) Conditional Operators/ Ternary Operators
- 7) Bitwise Operators

Arithmetic Operators:

An arithmetic operator performs mathematical operations such as addition (+), subtraction (-), multiplication (*), division (/) and modulus division (%) on numerical values (constants and variables).

S.no	Arithmetic Operators	Operation	Example
1	+	Addition	A+B
2	-	Subtraction	A-B
3	*	multiplication	A*B
4	/	Division	A/B
5	%	Modulus	A%B

//C Program to demonstrate the working of arithmetic operators

Compiled by Bibek Joshi

```

#include <stdio.h>
void main()
{
    int a = 10, b = 5, add, sub, mul, div, mod;
    add = a+b; // add =15
    printf("a+b = %d \n", add);
    sub = a-b; // sub=5
    printf("a-b = %d \n", sub);
    mul = a*b; // mul=50
    printf("a*b = %d \n", mul);
    div=a/b; // div= 2
    printf("a/b = %d \n", div);
    mod=a%b; // mod= 0
    printf("Remainder when a divided by b = %d \n", mod);
}

```

Relational Operators:

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0. Operands may be variables, constants or expressions.

Relational operators are used in decision-making and loops.

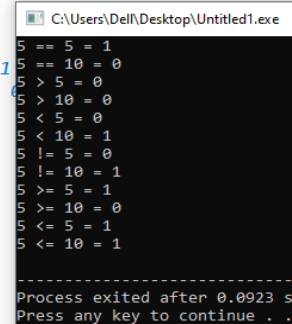
Operator	Meaning	Example	Return value
<	is less than	2<9	1
<=	is less than or equal to	2 <= 2	1
>	is greater than	2 > 9	0
>=	is greater than or equal to	3 >= 2	1
==	is equal to	2 == 3	0
!=	is not equal to	2!=2	0

// C Program to demonstrate the working of relational operators

```

#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;
    printf("%d == %d = %d \n", a, b, a == b); // true 1
    printf("%d == %d = %d \n", a, c, a == c); // false 0
    printf("%d > %d = %d \n", a, b, a > b); //false 0
    printf("%d > %d = %d \n", a, c, a > c); //false 0
    printf("%d < %d = %d \n", a, b, a < b); //false 0
    printf("%d < %d = %d \n", a, c, a < c); //true 1
    printf("%d != %d = %d \n", a, b, a != b); //false 0
    printf("%d != %d = %d \n", a, c, a != c); //true 1
    printf("%d >= %d = %d \n", a, b, a >= b); //true 1
    printf("%d >= %d = %d \n", a, c, a >= c); //false 0
    printf("%d <= %d = %d \n", a, b, a <= b); //true 1
    printf("%d <= %d = %d \n", a, c, a <= c); //true 1
    return 0;
}

```



```

C:\Users\Del\l\Desktop\Untitled1.exe
5 == 5 = 1
5 == 10 = 0
5 > 5 = 0
5 > 10 = 0
5 < 5 = 0
5 < 10 = 1
5 != 5 = 0
5 != 10 = 1
5 >= 5 = 1
5 >= 10 = 0
5 <= 5 = 1
5 <= 10 = 1
-----
Process exited after 0.0923 s
Press any key to continue . .

```

Logical operators :

Expressions that use logical operators are evaluated to be either true (1) or false (0). There are generally three logical operators used in C, Logical AND (&&), Logical OR (||) and Logical NOT (!).

Operator	Meaning	Example	Return value
&&	Logical AND	(9>2)&&(17>2)	1
	Logical OR	(9>2) (17 == 7)	1
!	Logical NOT	29!=29	0

// C Program to demonstrate the working of logical operators

```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) is %d \n", result);

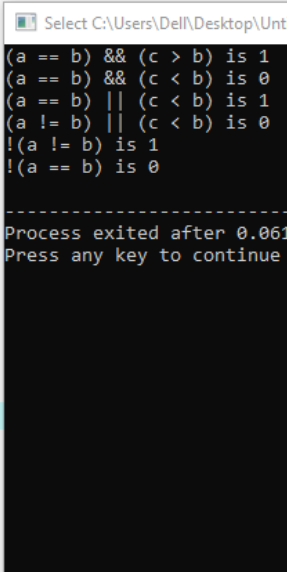
    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) is %d \n", result);

    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) is %d \n", result);

    result = !(a != b);
    printf("(a != b) is %d \n", result);

    result = !(a == b);
    printf("(a == b) is %d \n", result);

    return 0;
}
```



```
Select C:\Users\Del\\Desktop\Unti
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0
-----
Process exited after 0.061
Press any key to continue
```

Assignment Operator:

An assignment operator assigns a value to a variable and is represented by “=” (equals to) symbol.

Syntax: Variable name = expression.

The **Compound assignment operators** (+=, *=, -=, /=, %=) are also used.

Operator	Example	Meaning
+=	x += y	x=x+y
- =	x - = y	x=x-y
* =	x * = y	x=x*y
/ =	x / = y	x=x/y
% =	x % = y	X=x%y

// C Program to demonstrate the working of assignment operators

```
int main()
{
    int a = 5, c;
    c = a; //c=5
    printf("c = %d \n", c);
    c += a; // c = c+a //c=5+5=10
    printf("c = %d \n", c);
    c -= a; // c = c-a //c=10-5=5
    printf("c = %d \n", c);
    c *= a; // c = c*a //c=5*5=25
    printf("c = %d \n", c);
    c /= a; // c = c/a //c=25/5=5
    printf("c = %d \n", c);
    c %= a; // c = c%a //c=5%5=0
    printf("c = %d \n", c);
    return 0;
}
```

Unary Operators:

A operator acts up on a single operand to produce a new value is called a unary operator.

a. Increment/Decrement Operator:

The increment operator (++) increases its operand by 1 and the decrement (--) decreases its operand by 1. They are the unary operators (i.e. they operate on a single operand). It can be written as:

x++ or ++x (post and pre-increment)

x-- or --x (post and pre decrement)

Operator Meaning

++x Pre increment

--x Pre decrement

x++ Post increment

x-- Post decrement

Compiled by Bibek Joshi

Where

- ++x : Pre increment, first increment, and then do the operation.
- -x : Pre decrement, first decrements, and then do the operation.
- x++ : Post increment, first do the operation and then increment.
- x-- : Post decrement, first do the operation and then decrement.

```
#include <stdio.h>
int main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;
    //pre increment and decrement
    printf("++a = %d \n", ++a); // 11
    printf("--b = %d \n", --b); //99
    printf("++c = %f \n", ++c); //11.50
    printf("--d = %f \n", --d); // 99.50
    // Post increment and Decrement
    printf("a++ = %d \n", a++); //11
    printf("b-- = %d \n", b--); //99
    printf("c++ = %f \n", c++); //11.50
    printf("d-- = %f \n", d--); //99.50
    return 0;
}
```

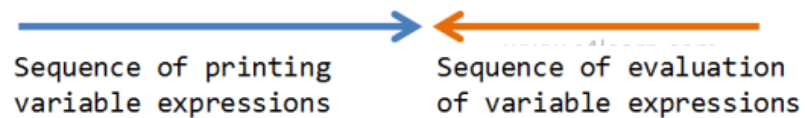
Multiple increment operators inside printf

```
#include<stdio.h>
void main() {
    int i = 1;
    printf("%d %d %d", i, ++i, i++);
}
```

Output : 3 3 1

Pictorial representation

```
printf("%d %d %d", i, ++i, i++);
```



Explanation of program

1. Whenever more than one format specifiers (i.e %d) are directly or indirectly related with same variable (i,i++,++i) then we need to evaluate each individual expression from right to left.
2. As shown in the above image evaluation sequence of expressions written inside printf will be – i++,++i,i
3. After execution we need to replace the output of the expression at the appropriate place

No	Step	Explanation
1	Evaluate i++	At the time of execution we will be using older value of i = 1
2	Evaluate ++i	At the time of execution we will be increment value already modified after step 1 i.e i = 3
2	Evaluate i	At the time of execution we will be using value of i modified in step 2

b. Sizeof Operator:

The **sizeof()** operator is commonly used in C. It determines the size of the expression or the data type specified in the number of char-sized storage units. When **sizeof()** is used with the data types, it simply returns the amount of memory allocated to that data type.

```
#include <stdio.h>
int main() {
int a = 20;
printf("Size of variable a : %d",sizeof(a)); //4
printf("\nSize of int data type : %d",sizeof(int)); //4
printf("\nSize of char data type : %d",sizeof(char)); //1
printf("\nSize of float data type : %d",sizeof(float)); //4
printf("\nSize of double data type : %d",sizeof(double)); //8
return 0;
}
```

Conditional Operator/ Ternary operator:

conditional operator checks the condition and executes the statement depending of the condition.

A conditional operator is a ternary operator, that is, it works on 3 operands.

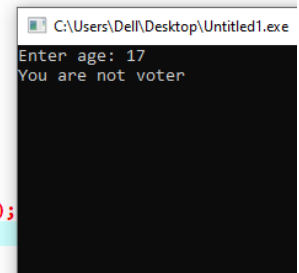
Conditional operator consist of two symbols.

1 : question mark (?).

2 : colon (:).

Syntax : condition ? exp1 : exp2;

```
//conditional operator example
#include<stdio.h>
#include<conio.h>
void main()
{
    int age;
    printf("Enter age: ");
    scanf("%d",&age);
    age>=18?printf("You are voter"):printf("You are not voter");
    getch();
}
```



Bitwise operators

These operators which are used for the manipulation of data at bit level are known as bitwise operators. These are used for testing the bits or shifting them right or left.

Operator	Meaning
&	Binary (bitwise) AND
	Bitwise OR
^	Bitwise exclusive OR
<<	Bitwise left shift
>>	Bitwise right shift
~	Bitwise Ones Complement

a. Bitwise AND operator

```
a = 1100; // (12 in decimal)
b = 1010; // (10 in decimal)
c = a & b; // (8 in decimal)
c = 1000;
```

b. Bitwise OR operator

```
a = 1100; // (12 in decimal)
b = 1010; // (10 in decimal)
c = a | b; // (14 in decimal)
c = 1110;
```

c. Exclusive OR (XOR) operator

```
c = a ^ b; // (6 in decimal)
c = 0110;
(Note: if both bits are same then zero otherwise one).
```

d. Bitwise Complement (~) operator

```
~a = ~12 = -13
(Note: bitwise Complement of N is - (N+1))
```

e. Left Shift operator:

$a \ll 1$
 $= 12 \ll 1$

Therefore $a \ll 1 = (11000)_2 = (24)_{10}$

Note: In left shift operator, first convert the given number into binary form and append the specified no of Zero's at the end. Then the resulting binary is convert in decimal form which is final answer.

f. Right Shift operator:

$a \gg 1$
 $= 12 \gg 1$

Therefore $a \gg 1 = (110)_2 = (6)_{10}$

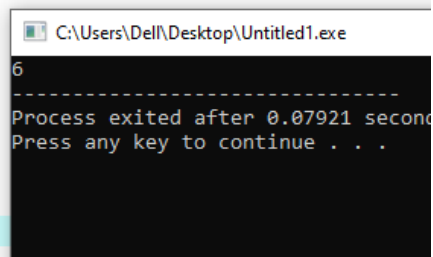
Note: In right shift operator, first convert the number given binary form and then remove the no. of specified bits from right side. Then the resulting binary number is Convert in decimal from which is final answer.

```
// C Program to demonstrate use of bitwise operators
#include <stdio.h>
int main()
{
    // a = 5(101), b = 6(110)
    int a = 5, b = 6;
    printf("a = %d, b = %d\n", a, b);
    // The result of a&b=100=4
    printf("a&b = %d\n", a & b);
    // The result of a|b=111=7
    printf("a|b = %d\n", a | b);
    // The result of a^b=011=3
    printf("a^b = %d\n", a ^ b);
    // The result is ~a= -(5+1)=-6 note: ~n=-(n+1)
    printf("~a = %d\n", a = ~a);
    /* The result of b<<1=1100=12 note: Left shift add 0
    at the end according to shif number*/
    printf("b<<1 = %d\n", b << 1);
    /* The result of b>>1=3 note: right shift remove no. of
    specified bit at the end according to shift number*/
    printf("b>>1 = %d\n", b >> 1);
    return 0;
}
```

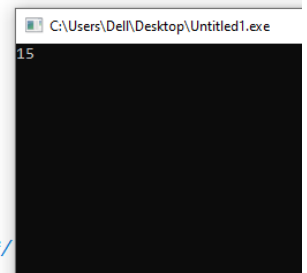

Comma Operator:

The comma in the form of an operator is used to assign multiple numbers of values to any variable in a program.

```
#include<stdio.h>
int main()
{
    int y;
    int a = (y=2,y+4);
    printf("%d", a);
    return 0;
}
```



```
#include <stdio.h>
int main()
{
    int x = 10;
    int y = 15;
    // using comma as an operator
    printf("%d", (x, y));
    getchar();
    return 0;
}
/*It evaluates the first operand & discards the result,
evaluates the second operand & returns the value as a result.*/
```



EXPRESSIONS:

- An expression is a sequence of operands and operators that reduces to a single value.
- Expression can be simple or complex.
- An operator is a syntactical token that requires an action be taken.
- An operand is an object on which an operation is performed.

A simple expression contains only one operator. E.g: $2 + 3$ is a simple expression whose value is 5. A complex expression contains more than one operator. E.g: $2 + 3 * 2$.

To evaluate a complex expression we reduce it to a series of simple expressions. In first we will evaluate the simple expression $3 * 2$ (6) and then the expression $2 + 6$, giving a result of 8.

The order in which the operators in a complex expression are evaluated is determined by a set of priorities known as precedence, the higher the precedence, the earlier the expression containing the operator is evaluated.

The following table shows the precedence and Associativity of operators:

Rules of Associativity

Operator	Associativity
() [] ->	Left to right
! ~ ++ -- - (type) * & sizeof()	Right to left
* / %	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
	Left to right
^	Left to right
&&	Left to right
	Left to right
?:	Right to left
= += -= /= %= &= = ^=	Right to left
,	Left to right

TYPE CONVERSION:

In an expression that involves two different data types, such as multiplying an integer and a floating-point number to perform these evaluations, one of the types must be converted.

We have two types of conversions:

1. Implicit Type Conversion
2. Explicit Type Conversion

IMPLICIT TYPE CONVERSION: When the types of the two operands in a binary expression are different automatically converts one type to another. This is known as implicit type conversion.

```
//implicit type conversion
#include <stdio.h>
main() {
    int number = 1;
    char character = 'k'; /*ASCII value is 107 */
    int sum;
    sum = number + character;
    printf("Value of sum : %d\n", sum );
}
```

C:\Users\Del\\Desktop\Untitled1.
Value of sum : 108

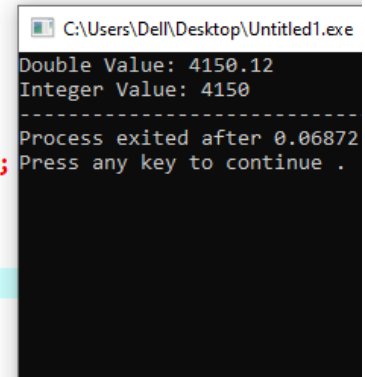
Compiled by Bibek Joshi

```
//implicit type conversion
#include<stdio.h>

int main() {

    // create a double variable
    double value = 4150.12;
    printf("Double Value: %.2lf\n", value);

    // convert double value to integer
    int number = value;
    printf("Integer Value: %d", number);
    return 0;
}
```



```
C:\Users\Del\Desktop\Untitled1.exe
Double Value: 4150.12
Integer Value: 4150
-----
Process exited after 0.06872
Press any key to continue .
```

EXPLICIT TYPE CONVERSION: Explicit type conversion uses the unary cast operator to convert data from one type to another. To cast data from one type to another, we specify the new type in parentheses before the value we want converted.

Syntax: (data type) variable/expression/value;

Example:

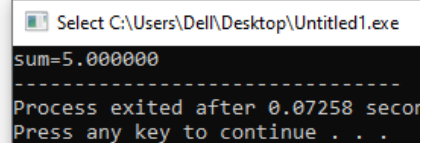
```
float sum;
```

```
Sum = (int) (1.5 * 3.8);
```

The typecast (int) tells the C compiler to interpret the result of (1.5 * 3.8) as the integer 5, instead of 5.7.

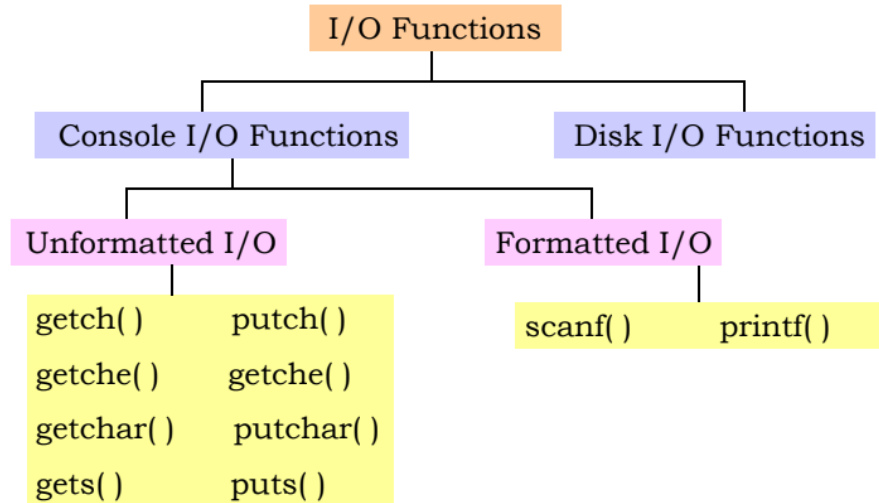
Then, 5.0 will be stored in sum, because the variable sum is of type float.

```
//Explicit Type Conversion
#include <stdio.h>
int main()
{
    float sum;
    sum = (int) (1.5 * 3.8);
    printf("sum=%f", sum);
}
```

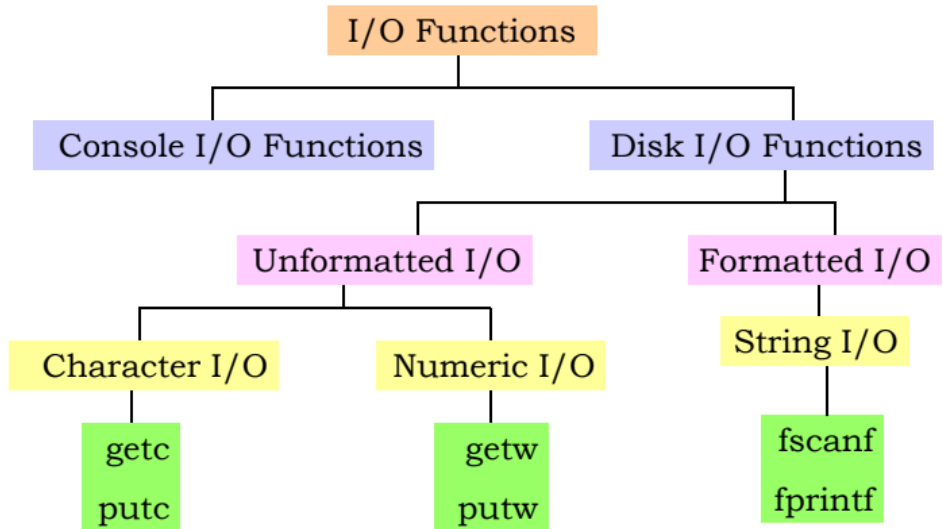


```
Select C:\Users\Del\Desktop\Untitled1.exe
sum=5.000000
-----
Process exited after 0.07258 second
Press any key to continue . . .
```

Console Input/Output Functions



Disk Input/Output Functions



Formatted Input Function

scanf()

- ▶ `int scanf(char *format, args...)` -- reads from stdin and puts input in address c variables specified in argument list.
- ▶ Returns, number of characters read.
- ▶ The address of variable or a pointer to one is required by `scanf`.
Example: `scanf("%d",&i);`
- ▶ We can just give the name of an array or string to `scanf` since this corresponds to the start address of the array/string.

Example:

```
char string[80];  
scanf("%s",string);
```

printf()

- The `printf` function is defined as follows:
`int printf(char *format, arg list ...)` --
prints to stdout the list of arguments according specified format string.
Returns number of characters printed.
- The format string has 2 types of object:
- *ordinary characters* -- these are copied to output.
- *conversion specifications* -- denoted by `%` and listed in Table.

Program illustrating Formatted I/O Functions

```
#include <stdio.h>

int main()
{
    int day;
    int month;
    int year;
    char name[30];

    printf("Enter your name:\n");
    scanf("%s", name);

    /* skipping spaces */
    printf("Hi %s. Enter birthdate as: dd mm yyyy\n", name);
    scanf("%d %d %d", &day, &month, &year);

    /* alternative */
    printf("Hi %s. Enter birthdate as: dd-mm-yyyy\n", name);
    scanf("%d-%d-%d", &day, &month, &year);

    return 0;
}
```

Note: no ampersand for strings

Conversion specifier

Literal characters

Unformatted Console I/O Functions

getch() and getche()

- ▶ This function will read a single character the instant it is typed without waiting for the Enter key to be hit.
- ▶ These functions return the character that has been more recently typed.
- ▶ The 'e' in *getche()* function means it echoes(displays) the character that has been typed.
- ▶ The *getch()* just return the character that has been typed without echoing it on the screen.

Unformatted Console I/O Functions

`getchar()` and `putchar()`

- ▶ The *`getchar()`* accepts a single character the instant it has been typed.
- ▶ The *`getchar()`* echoes the character that has been typed.
- ▶ It requires the Enter key to be typed following the character that has been typed.
- ▶ The functions *`putch()`* and *`putchar()`* print the character on the screen.

Program illustrating Console Input Functions

```
main()  
{  
    char ch;  
    printf("Hello\n");  
    getch();  
    printf("Type any character\n");  
    ch=getche();  
    printf("Type any character\n");  
    ch=getchar();  
}
```

Output:

Hello

Type any character A

Type any character B

```
main()    Program illustrating Console Output Functions
{
    char ch='D';
    putchar(ch);
    putchar(ch);
    putchar('S');
    putchar('S');
}
```

Output:

DDSS

Iterations

Control statements for decision-making:

A control statement enables us to specify the order in which the various instructions in a program are to be executed by the computer. In other words, the control instructions determine the "flow of control" in a program. Control statements are classified in the following ways:

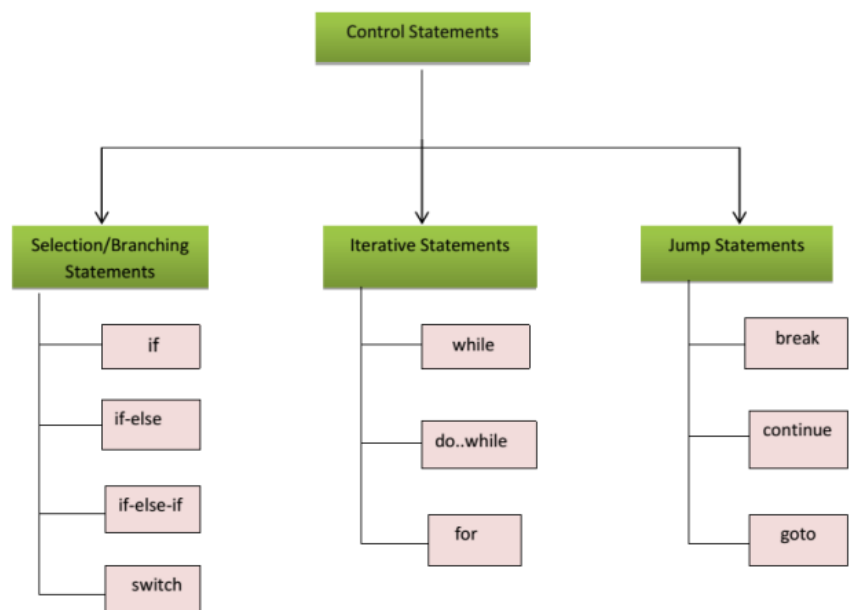
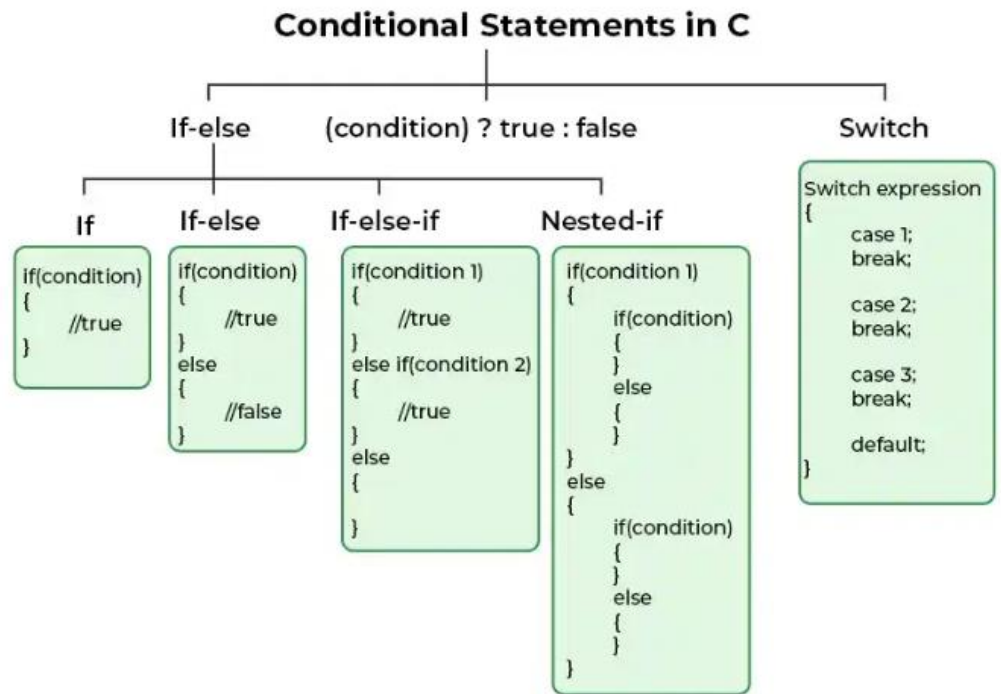


Fig: 1 Classification of control statements

Selection or Branching or Decision Control Instruction (Condition Expression)

The statements that make the selection of statements to be executed are called selection statements or branching statements. The selective structure consists of a test for a condition followed by an alternative path that the program can follow. The conditional expression is mainly used for decision-making. There are various structures of the control statements. They are:



if statement:

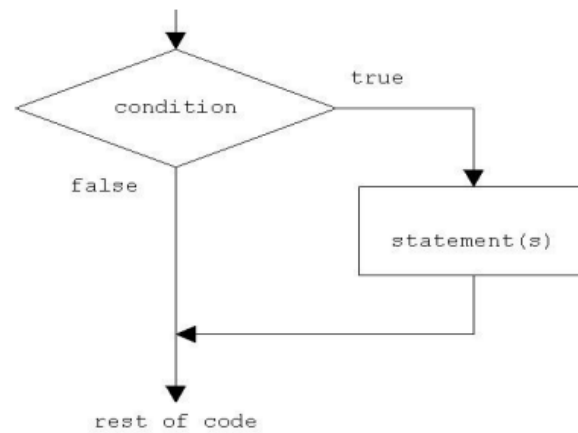
The **if** statement is used to express conditional expressions. If the given condition is true then it will execute the statements; otherwise, it will execute optional statements. The braces { and } are used to group declarations and statements into compound statements or block.

Syntax:

```
if(condition) {
//set of stmt1
}
```

```
//set of stmt2
```

If the condition is true set of stmt1 is executed and then after only the set of stmt2 is executed otherwise set of stmt1 is skipped and the direct set of stmt2 is executed.



Example:

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5      int a=15,b=10;
6      if(a>b)
7      {
8          printf("a is greater");
9      }
10     getch();
11 }
12
```

C:\Users\Dell\Desktop\Untitled1.exe
a is greater

```
Untitled1.c
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5      int num;
6      printf("enter any number: ");
7      scanf("%d",&num);
8      if(num<0)
9      {
10         printf("you entered number %d \n",num);
11     }
12     if(num>0)
13     {
14         printf("you entered number %d \n",num);
15     }
16     getch();
17 }
18
```

C:\Users\Dell\Desktop\Untitled1.exe

enter any number: -7
you entered number -7

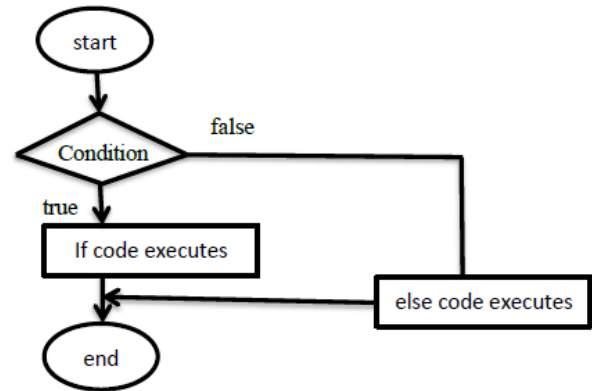
If-else statement:

The if-else statement is an extension of the simple if statement. The general form is. The if...else statement executes some code if the test expression is true otherwise the else statement is executed.

Syntax :

```
if (condition)
{
    true statement;
}
else
{
    false statement;
}
```

Flowchart



Example:

```
20 // Program to check whether an integer entered by the user is odd or even
21 #include <stdio.h>
22 int main()
23 {
24     int number;
25     printf("Enter an integer: ");
26     scanf("%d",&number);
27     if( number%2 == 0 )
28         printf("%d is an even integer.",number);
29     else
30         printf("%d is an odd integer.",number);
31     return 0;
32 }
```

```
33 C:\Users\Dell\Desktop\Untitled1.exe
Enter an integer: 5
5 is an odd integer.
-----
Process exited after 2.489 seconds with return value 0
Press any key to continue . . .
```

```

1 //A program to read any two numbers from the keyboard and to display the larger value.
2 #include <stdio.h>
3 #include <conio.h>
4 void main()
5 {
6     float x,y;
7     printf("Enter first number:");
8     scanf("%f",&x);
9     printf("Enter second number:");
10    scanf("%f",&y);
11    if(x>y)
12        printf("Larger value is %f",x);
13    else
14        printf("Larger value is %f",y);
15    getch();
16 }

```

C:\Users\Del\\Desktop\Untitled1.exe

```

Enter first number:5
Enter second number:4
Larger value is 5.000000

```

```

1 //Write a program to check whether a given year is Leap year or not
2 #include <stdio.h>
3 int main()
4 {
5     int year;
6     printf("Enter a year to check if it is a leap year\n");
7     scanf("%d", &year);
8     if (year%4 == 0) && ((year%100 != 0) || (year%400 == 0))
9         printf("%d is a leap year.\n", year);
10    else
11        printf("%d is not a leap year.\n", year);
12    return 0;
13 }

```

C:\Users\Del\Desktop\Untitled1.exe

```

Enter a year to check if it is a leap year
2080
2080 is a leap year.

```

Nested if-else statement:

When a series of decisions are involved, we may have to use more than one if-else statement in nested form. If-else statements can also be nested inside another if block or else block or both.

Syntax:

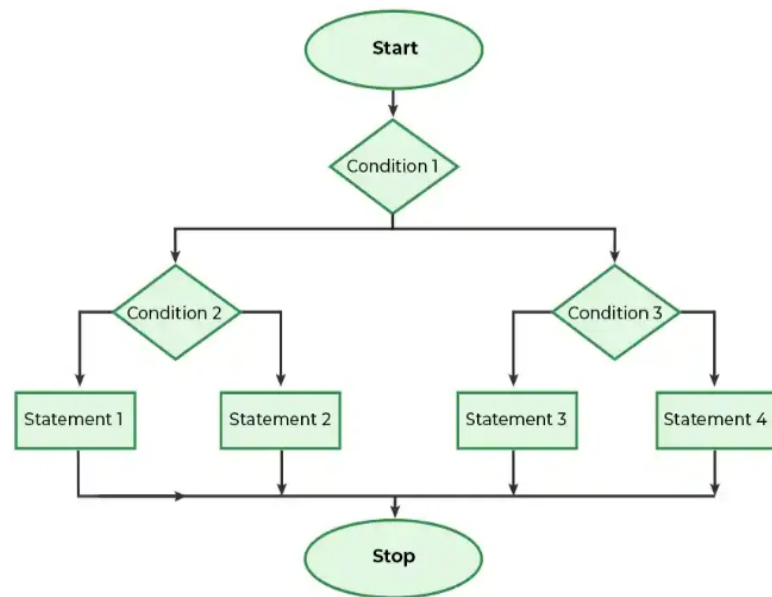
```

if (expression)
{
    if(expression)

```

Compiled by Bibek Joshi

```
    else
    {
    }
}
else
{
    if(expression)
    {
    }
    else
    {
    }
}
```



Example:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int age;
6      printf("Please Enter Your Age Here:\n");
7      scanf("%d",&age);
8      if ( age < 18 )
9      {
10         printf("You are Minor.\n");
11         printf("Not Eligible to Work");
12     }
13     else
14     {
15         if (age >= 18 && age <= 60 )
16         {
17             printf("You are Eligible to Work \n");
18             printf("Please fill in your details and apply\n");
19         }
20     }
21     {
22         printf("You are too old to work as per the Government rules\n");
23         printf("Please Collect your pension! \n");
24     }
25 }
26 return 0;
27 }
```

C:\Users\Dell\Desktop\Untitled1.exe

```
Please Enter Your Age Here:
25
You are Eligible to Work
Please fill in your details and apply
```

Example 2:

WAP to find the largest number among 3 numbers.


```
1  #include <stdio.h>
2  #include <conio.h>
3  void main()
4  {
5      float x,y,z;
6      printf("Enter first number:");
7      scanf("%f",&x);
8      printf("Enter second number:");
9      scanf("%f",&y);
10     printf("Enter third number:");
11     scanf("%f",&z);
12     if(x>z)
13     {
14         if(x>y)
15             printf("Largest value is %f",x);
16         else
17             printf("Largest value is %f",y);
18     }
19     else
20     {
21         if(z>y)
22             printf("Largest value is %f",z);
23         else
24             printf("Largest value is %f",y);
25     }
26     getch();
27 }
28
```

Select C:\Users\Dell\Desktop\Untitled1.exe

```
Enter first number:9
Enter second number:5
Enter third number:7
Largest value is 9.000000
```

Else if ladder

The if else-if statement is used to execute one code from multiple conditions. If any of these expressions is formed to be true, the statement associated with it is executed and this terminates the whole chain. If none of the expressions is true then the statement associated with the final else is executed.

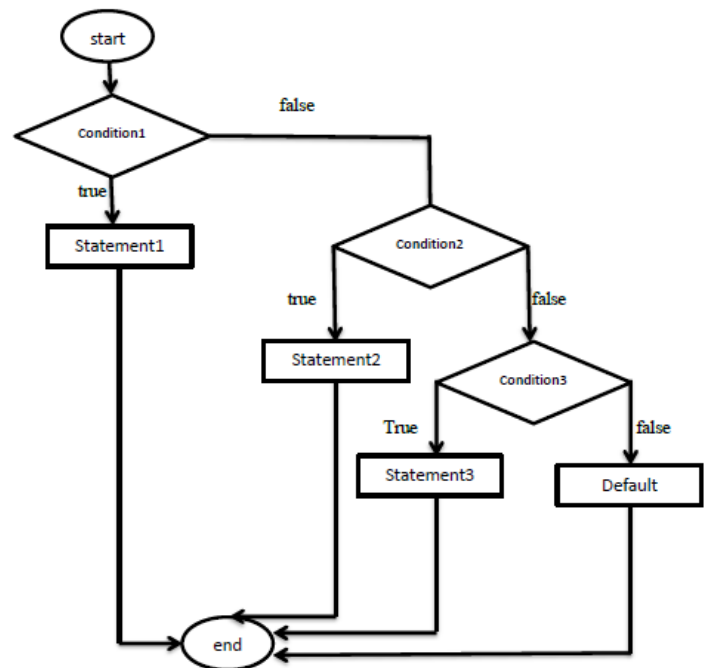
Syntax : if(condition-1)

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

Compiled by Bibek Joshi

```
statement-3;  
}  
else if(condition-n)  
{  
statement-n;  
}  
else  
{  
default-statement;  
}
```

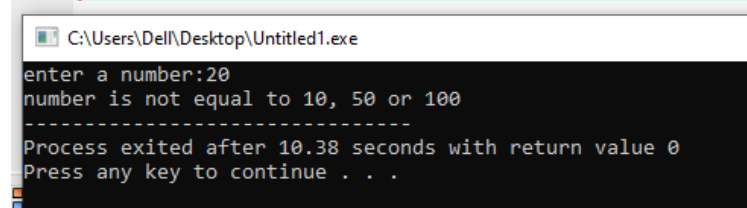
Flowchart



Example 1

WAP to check whether the entered number is 10,50,100 or not.

```
1  #include<stdio.h>
2  int main(){
3      int number=0;
4      printf("enter a number:");
5      scanf("%d",&number);
6      if(number==10){
7          printf("number is equals to 10");
8      }
9      else if(number==50){
10         printf("number is equal to 50");
11     }
12     else if(number==100){
13         printf("number is equal to 100");
14     }
15     else{
16         printf("number is not equal to 10, 50 or 100");
17     }
18     return 0;
19 }
```



Example 2: WAP to print weekday based on given number.

```
#include<stdio.h>

int main()
{
    int day;
    printf("Enter day number: ");
    scanf("%d", &day);
    if(day==1)
    {
        printf("SUNDAY.");
    }
    else if(day==2)
    {
        printf("MONDAY.");
    }
}
```

Compiled by Bibek Joshi

```

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 DAY.");
}
return (0);
}

```

Example 3:

```

#include<stdio.h>
#include <conio.h>
void main()
{
    int marks;
    printf("Enter marks of a student\n");
    scanf("%d",&marks);
    if(marks>=80)
    {
        printf("Student is passed in distinction:\n");
    }
    else if(marks>=60)
    {
        printf("Student is passed in first division:\n");
    }
    else if(marks>=45)
    {
        printf("Student is passed in second division:\n");
    }
}

```

Compiled by Bibek Joshi

```

else if(marks>=32)
{
printf("Student is passed in third division:\n");
}
else
{
printf("Student is failed:\n");
}
getch();
}

```

Example 4:

An electricity board charges according to following rates.

For the first 100 units-----Rs. 40 per unit

For the next 200 units-----Rs. 50 per unit Beyond 300 units-----Rs. 60 per unit

All users are charged meter charge also, which is also Rs. 50. Write a program to read the number of units consumed, and print out the charges.

```

#include<stdio.h>
#include<conio.h>
#define meter_charge 50
void main()
{
int units, totalcharge, charge;
clrscr();
printf("enter units of electricity used");
scanf("%d",&units);
if(units>300)
charge =(units-300)*60+200*50+100*40;
else if(units>100)
charge =(units-100)*50+100*40;
else
charge =units*40;
totalcharge =charge + meter_charge;
printf("\nThe payable amount is %d",totalcharge);
getch();
}

```

Alternate method:

```

#include<stdio.h>
#include<conio.h>
#define meter_charge 50
void main()
{
int units;
int charge, totalcharge;
printf("enter units of electricity used"); scanf("%d",&units);
if(units<=100)
charge=100*40;
else if(units<=300)
charge=(units-100)*50+100*40;
else

```

Compiled by Bibek Joshi

```
charge=(units-300)*60+200*50+100*40;
totalcharge=charge + meter_charge;
printf("\nThe payable amount is %d",totalcharge);
getch();
clrscr();
}
```

Switch Statement

- The control statement that allows us to make a decision from the number of choices is called a **switch**, or more correctly a **switch-case-default**
- Syntax

```
switch (expression)
{
    case value 1 :
        do this ;
        break;
    case value 2 :
        do this ;
        break;
    case value 3 :
        do this ;
        break;
    default :
        do this ;
}
```

switch (expression) ← Expression is integer or character

case value 1 : ← case ends with colon (:) { do this ; break; } ← Case Block

The value of expression is match against case value

The break statement when used in a switch takes the control outside the switch

Statement-x;

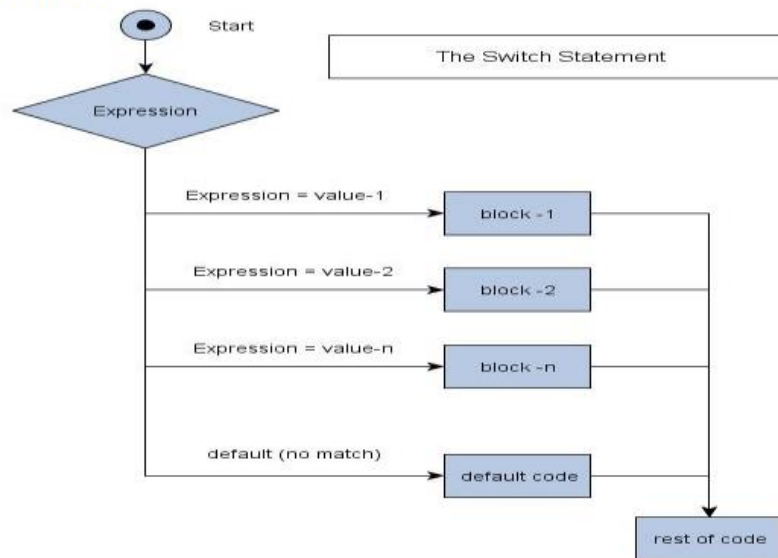
Switch Statement

```
char single;
printf("Enter a character");
scanf("%c",& single);
switch(single)
{
    case 'x' :
        printf("x");
        break;
    case 'y' :
        printf("y");
        break;
    case 'z' :
        printf("z");
        break;
    default:
        printf("Not x and y and z \n");
}
printf("done");
```

- Above example is equivalent to following if-else-if statement

```
char single;  
printf("Enter a character");  
scanf("%c",& single);  
  
if(single=='x')  
    printf("x");  
else if(single=='y')  
    printf("y");  
else if(single=='z')  
    printf("z");  
else  
    printf("Not x and y and z \n");  
  
printf("done");
```

Flowchart



Example1:

WAP to determine whether a user entered number is odd or even.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a;
    clrscr();
    printf("Enter any number\n");
    scanf("%d",&a);
    a=a%2;
    switch(a)
    {
        case 0:
            printf("The number is even\n");
            break;
        default:
            printf("The number is odd");
    }
    getch();
}
```

Example2:

WAP to find the day on entering a number less than or equal to 7.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int day; clrscr();
    printf("Enter numeric day of the week:\n");
    scanf("%d",&day);
    switch(day)
    {
        case 1:
            printf("Day is Sunday\n");
            break;
        case 2:
            printf("Day is Monday\n");
            break;
        case 3:
            printf("Day is Tuesday\n");
            break;
        case 4:
            printf("Day is Wednesday\n");
            break;
    }
}
```

Compiled by Bibek Joshi

```

case 5:
printf("Day is Thursday\n");
break;
case 6:
printf("Day is Friday\n");
break;
case 7:
printf("Day is Saturday\n");
break;
default:
printf("Your choice is wrong!!!!!!");
break;
}
getch();
}

```

Example3:

Write a program to calculate area or perimeter of a rectangle.

```

#include<stdio.h>
#include<conio.h> void main()
{
float length, breadth, Area, P; int choice;
clrscr();
printf("Enter the value of length & breadth:");
scanf("%f %f",&length,&breadth);
printf("1> Area\n");
printf("2> Perimeter\n");
printf("Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
Area=length*breadth;
printf("The area is %f",Area);
break;
case 2:
P=2*(length+breadth);
printf("The Perimeter is %f", P);
break;
}
getch();
}

```

```

1 //Write a program to find whether a input char is vowel or constant or other characters using switch statement.
2 #include <stdio.h>
3 int main() {
4     char c;
5     printf("Enter a character:");
6     scanf("%c", &c);
7     if(c>='A'&&c<='Z' || c>='a'&&c<='z')
8     {
9         switch (c) {
10            case 'a':
11            case 'e':
12            case 'i':
13            case 'o':
14            case 'u':
15            case 'A':
16            case 'E':
17            case 'I':
18            case 'O':
19            case 'U':
20                printf("It is vowel");
21                break;
22            default:
23                printf("It is consonant");
24            }
25        }
26    else
27        printf("It is other character");
28    return 0;
29 }

```

Select C:\Users\Del\\Desktop\Untitled1.exe

```

Enter a character:e
It is vowel
-----

```

```

1 //Write a program to find whether given an integer is divisible by 3 and 5 but not by 10.
2 #include <stdio.h>
3 #include <conio.h>
4 main( )
5 {
6     int n;
7     printf("Enter an integer:");
8     scanf("%d", &n);
9     if(n%3 == 0 && n%5 == 0)
10    {
11        if(n%10!=0)
12        {
13            printf("%d is divisible by 3 & 5 and not by 10", n);
14        }
15    }
16    else
17    {
18        printf("%d is divisible by 3, 5 & 10", n);
19    }
20 }

```

C:\Users\Del\Desktop\Untitled1.exe

```

Enter an integer:30
30 is divisible by 3, 5 & 10
-----
Process exited after 2.942 seconds with return value 28
Press any key to continue . . .

```

Compiled by Bibek Joshi

Loop Statements or Iteration:

Loops are used when we want to execute a part of a program or block of statements several times. So, a loop may be defined as a block of statements that are repeatedly executed for a certain number of times or until a particular condition is satisfied. There are three types of loop statements in C:

1. For
2. While
3. Do...while

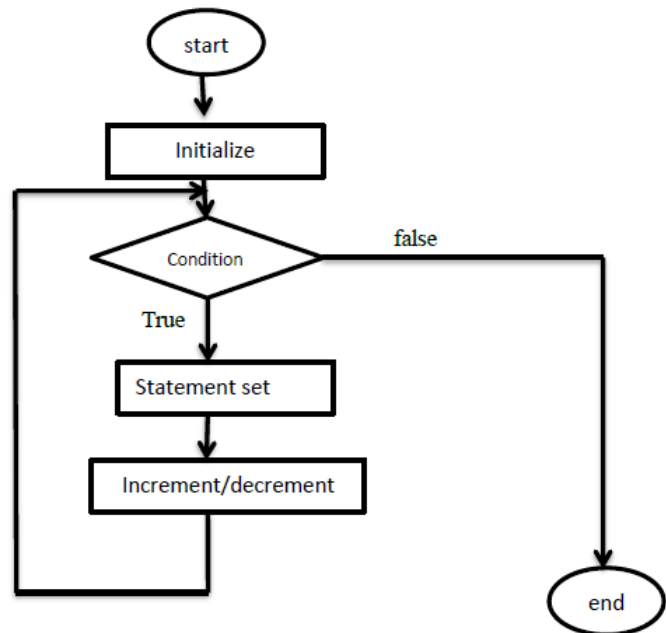
For Loop:

For loop is the most popular looping instruction. This is an **entry-controlled** looping statement because it checks the condition at the entry point. One of the most important features of this loop is that three actions can be taken at a time like variable initialization, condition checking, and increment/decrement. The for loop can be more concise and flexible than that of while and do-while loops.

Syntax:

```
for (initialization; condition; increment/decrement)
{
    Statements;
}
```

Flowchart



```

1 //WAP to Display N natural numbers.
2 #include <stdio.h>
3 int main()
4 {
5     int n,i;
6     printf("Enter the limit: ");
7     scanf("%d", &n);
8     for(i=1;i<=n;i++)
9     {
10    printf(" \t%d",i);
11    public int __cdecl printf (const char * __restrict__ _Format, ...)
12    }
13 }

```

C:\Users\Devi\Desktop\Untitled1.exe

```

Enter the limit: 10
      1      2      3      4      5      6      7      8      9      10
-----
Process exited after 2.169 seconds with return value 10
Press any key to continue . . .

```

```

1 //WAP to print the sum of 1st N natural numbers.
2 #include <stdio.h>
3 int main()
4 {
5     int n,i,sum=0;
6     printf("Enter the limit: ");
7     scanf("%d", &n);
8     for(i=1;i<=n;i++)
9     {
10    sum = sum +i;
11    }
12    printf("Sum of N natural numbers is: %d",sum);
13 }

```

C:\Users\Devi\Desktop\Untitled1.exe

```

Enter the limit: 9
Sum of N natural numbers is: 45
-----
Process exited after 3.663 seconds with return value 31
Press any key to continue . . .

```

```

1 //Consider the following program
2 #include<stdio.h>
3 int main(void)
4 {
5     int num;
6     printf("    n n_cubed\n");
7     for (num = 1; num <= 6; num++)
8         printf("%5d %5d\n", num, num*num*num);
9     return 0;
10 }

```

C:\Users\Dell\Desktop\Untitled1.exe

```

n n_cubed
1      1
2      8
3     27
4     64
5    125
6    216
-----
Process exited after 0.04831 seconds with return value 0
Press any key to continue . . .

```

```

1 //WAP to display factorial of n nmuber
2 #include<stdio.h>
3 #include<conio.h>
4 void main()
5 {
6     int i,fact=1,number;
7     printf("Enter a number: ");
8     scanf("%d",&number);
9     for(i=1;i<=number;i++){
10        fact=fact*i;
11    }
12    printf("Factorial of %d is: %d",number,fact);
13    getch();
14 }

```

C:\Users\Dell\Desktop\Untitled1.exe

```

Enter a number: 5
Factorial of 5 is: 120

```

Example:

WAP to print all even numbers from 0 to 10.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i=0;
    for ( ;i<=10; )
    {
        printf("%d\t",i);
        i=i+2;
    }
    getch();
}
```

Example:

WAP to count all the integers greater than 100 and less than 200 that are divisible by 7. Also, find the sum of these numbers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, count=0, sum=0;
    for(i=101; i<200;i++)
    {
        if (i%7 == 0)
        {
            sum = sum + i;
            count = count + 1;
        }
    }
    printf("The sum is = %d\n", sum);
    printf("The count is = %d", count);
    getch();
}
```

Example:

WAP to create the multiplication table of a number.

```
#include <stdio.h>
#include <conio.h>
void main ()
{
    int num,i,mul;
    printf("Enter a number:");
    scanf("%d",&num);
    for(i=1; i<=10;i++)
    {
        mul=num*i;
        printf("\n%d * %d=%d",num,i,mul);
    }
    getch();
}
```

Compiled by Bibek Joshi

```
}
```

Example:

//WAP to generate all character and their ASCII value.

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

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
    int i;
```

```
    for(i=0;i<=255;i++)
```

```
    {
```

```
        printf("%d = %c\t",i,i);
```

```
    }
```

```
    getch();
```

```
    return(0);
```

```
}
```

Various forms of FOR LOOP

I am using variable num in all the below examples

– 1) Here instead of num++, I'm using num=num+1 which is nothing but same as num++.

```
for (num=10; num<20; num=num+1)
```

2) Initialization part can be skipped from loop as shown below, the counter variable is declared before the loop itself.

```
int num=10;
```

```
for (;num<20;num++)
```

Must Note: Although we can skip init part but semicolon (;) before condition is must, without which you will get compilation error.

3) Like initialization, you can also skip the increment part as we did below. In this case semicolon (;) is must, after condition logic. The increment part is being done in for loop body itself.

```
for (num=10; num<20; )
```

```
{
```

```
//Code
```

```
num++;
```

```
}
```

4) Below case is also possible, increment in body and init during declaration of counter variable.

```
int num=10;
```

```
for (;num<20;)
```

```
{
```

```
//Statements
```

```
num++;
```

```
}
```

5) Counter can be decremented also, In the below example the variable gets decremented each time the loop runs until the condition num>10 becomes false.

```
for(num=20; num>10; num--)
```

Compiled by Bibek Joshi

While Loop:

While loop first checks whether the initial condition is true or false and finds it to be true, it will enter the loop and execute the statement. Thus, it is also an entry control loop (perform pre-test).

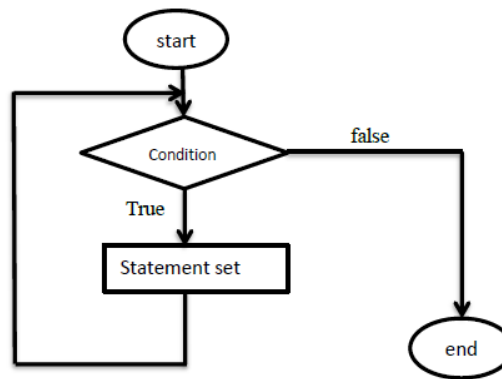
Syntax:

```
variable initialization;  
while (condition)  
{  
    statements;  
    variable increment or decrement;  
}
```

while loop can be addressed as an entry control loop. It is completed in 3 steps.

- Variable initialization. (e.g int x=0;)
- condition (e.g while(x<=10))
- Variable increment or decrement (x++ or x-- or x=x+2)

The while loop is an entry-controlled loop statement, i.e means the condition is evaluated first and if it is true, then the body of the loop is executed. After executing the body of the loop, the condition is once again evaluated and if it is true, the body is executed once again, the process of repeated execution of the loop continues until the condition finally becomes false and the control is transferred out of the loop.

Flowchart

```

1 //Example : Program to print first 10 natural numbers
2 #include<stdio.h>
3 #include<conio.h>
4 void main( )
5 {
6     int x;
7     x=1;
8     while(x<=10)
9     {
10         printf("%d\t", x);
11         x++;
12     }
13     getch();
14 }
15

```

C:\Users\Dell\Desktop\Untitled1.exe

```

1      2      3      4      5      6      7      8      9      10

```

```

1 //WAP to reverse a number
2 #include<stdio.h>
3 #include<conio.h>
4 main()
5 {
6     int n, reverse=0, rem;
7     printf("Enter a number: ");
8     scanf("%d", &n);
9     while(n!=0) //or while(n>0)
10    {
11        rem=n%10;
12        reverse=reverse*10+rem;
13        n/=10;
14    }
15    printf("Reversed Number: %d", reverse);
16    getch();
17 }

```

C:\Users\Dell\Desktop\Untitled1.exe

```

Enter a number: 345
Reversed Number: 543

```

```
1 //Write a program to find factorial using while loop.
2 #include<stdio.h>
3 #include<conio.h>
4 void main()
5 {
6     int fact=1, i=1,n;
7     printf("Enter the number:");
8     scanf("%d",&n);
9     while(i<=n)
10    {
11        fact=fact*i;
12        i++;
13    }
14    printf("Factorial of %d is %d",n,fact);
15    getch();
16 }
```

C:\Users\Del\\Desktop\Untitled1.exe

Enter the number:5
Factorial of 5 is 120

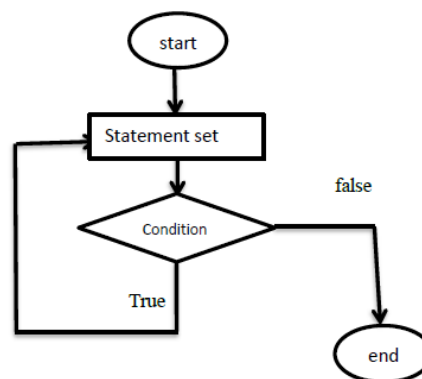
do-while loop:

The do-while loop is an exit-controlled loop statement. The body of the loop is executed first and then the condition is evaluated. If it is true, then the body of the loop is executed once again. The process of execution of the body of the loop is continued until the condition finally becomes false and the control is transferred to the statement immediately after the loop. The statements are always executed at least once.

Syntax:

```
variable initialization;
do {
    statements;
    variable increment or decrement;
} while (condition);
```

Flowchart



Compiled by Bibek Joshi

```

1 //Write a program to print the even number upto n using do-while loop.
2 #include<stdio.h>
3 #include<conio.h>
4 void main()
5 {
6     int n, i;
7     printf("Enter a value of n:");
8     scanf("%d",&n);
9     i=1;
10    do
11    {
12        if(i%2==0)
13        {
14            printf("%d\t",i);
15        }
16        i++;
17    }while(i<=n);
18    getch();
19 }

```

C:\Users\Del\\Desktop\Untitled1.exe

```

Enter a value of n:10
2      4      6      8      10

```

```

1 //Example : Program to print first ten multiple of 5
2 #include<stdio.h>
3 #include<conio.h>
4 void main()
5 {
6     int a,i;
7     a=5;
8     i=1;
9     do
10    {
11        printf("%d\t",a*i);
12        i++;
13    }while(i <= 10);
14    getch();
15 }

```

C:\Users\Del\\Desktop\Untitled1.exe

```

5      10     15     20     25     30     35     40     45     50

```

Nested Loop:

C programming language allows using one loop inside another loop. The following section shows a few examples to illustrate the concept.

Syntax:

The syntax for a nested for-loop statement in C is as follows:

```

for ( init; condition; increment )
{
for ( init; condition; increment)

```

Compiled by Bibek Joshi

```

{
    statement(s);
}
statement(s);
}

```

For example:

```

for(i=0;i<10;i++)
{
    for(j=0;j<10;j++) //nested loop
    {
        .....
    }
}

```

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

```

while(condition)
{
    while(condition)
    {
        statement(s);
    }
    statement(s);
}

```

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

```

do
{
    statement(s);
do
{
    statement(s);
}while( condition );

```

A final note on loop nesting is that you can put any type of loop inside of any other type of loop. For example, a for loop can be inside a while loop or vice versa.

```
1 #include <stdio.h>
2 int main()
3 {
4     int i,j,n; // variable declaration
5     printf("Enter the value of n :");
6     // Displaying the n tables.
7     scanf("%d",&n);
8     for(i=1;i<=n;i++) // outer loop
9     {
10         for(j=1;j<=10;j++) // inner loop
11         {
12             printf("%d\t",(i*j)); // printing the value.
13         }
14         printf("\n");
15     }
16     return 0;
17 }
```

C:\Users\Del\\Desktop\Untitled1.exe

Enter the value of n :2

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20

Process exited after 1.934 seconds with return value 0

Press any key to continue . . .

```
1 //Nested while loop example
2 #include <stdio.h>
3 int main()
4 {
5     int a = 1, b = 1;
6     while(a <= 5)
7     {
8         b = 1;
9         while(b <= 5)
10        {
11            printf("%d ", b);
12            b++;
13        }
14        printf("\n");
15        a++;
16    }
17    return 0;
18 }
```

C:\Users\Del\\Desktop\Untitled1.exe

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

Process exited after 0.05166 seconds with return value 0

Press any key to continue . . .

Compiled by Bibek Joshi

```

1 //WAP using a nested for loop to find the prime numbers from 2 to 20:
2 #include <stdio.h>
3 int main ()
4 {
5     /* local variable definition */
6     int i, j;
7     for(i=2; i<20; i++)
8     {
9         for(j=2; j <= (i/j); j++)
10            if(!(i%j))
11                break; // if factor found, not prime
12            if(j > (i/j)) printf("%d is prime\n", i);
13    }
14    return 0;
15 }

```

C:\Users\Del\\Desktop\Untitled1.exe

```

2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
-----
Process exited after 0.05134 seconds with return value 0
Press any key to continue . . .

```

```

/*printing the pattern
*****
*****
*****
***** */
#include <stdio.h>
int main()
{
    int i=1;
    do          // outer loop
    {
        int j=1;
        do      // inner loop
        {
            printf("*");
            j++;
        }while(j<=8);
        printf("\n");
        i++;
    }while(i<=4);
}

```

Select C:\Users\Del\\Desktop\Untitled1.exe

```

*****
*****
*****
*****
-----
Process exited after 0.06754 seconds with return value 10
Press any key to continue . . .

```

Continue statements

Continue statement is used to bypass the execution of the statements inside the loop. In some situations we want to jump to the next iterations of the loop by ignoring the execution of statements specified within the body of loop. The keyword “continue” allows us to do this. When the keyword is encountered inside the loop, control automatically passes to the next iteration of the loop. A continue is specified using if() statement.

Example:

```

void main() {
    int i;
    for (i =1; i<=5;i++)//The continue statement is always inside loop {
        if ( i = =3)
        {continue;
        }
        printf("%d",i); }

```

Compiled by Bibek Joshi


```
}
```

Output: 1 2 4 5

Thus, continue statement is used to skip a part of the body of loop and continue with next iteration. Note that when the value of i equal to 3, the continue statement takes the control to the next iteration bypassing the execution of the printf() statement.

Break Statement

The break statement is used as a statement inside the body of loop. Generally the break statement is associated with a condition specified with if () statement. As soon as the condition evaluates within if (), the execution of break statement takes the control outside the loop and therefore stops the further iterations of the loop.

Example:

```
void main()
{
    int i;
    for (i =1; i<=5;i++)
    {
        printf("%d",i);
        if ( i ==3)
            break;
    }
    printf("\n loop terminates here"); }
```

Output:

1 2 3

Loop terminates here

Q. What is the similarity and difference between break and continue statements?

Similarity:

Both break and continue are jump statements.

Difference: The break statement terminates the entire loop execution whereas continue statement terminates single pass of the loop.

goto statement

The goto statement is used to alter the program execution sequence by transferring the control to some other part of the program. It is used to jump to a specific location.

Example:

Program to demonstrate the usage of goto statement.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a,b;
    printf("Enter two numbers:");
    scanf("%d %d",&a,&b);
    if(a>b)
        goto label1;
    else
        goto label2;
label1:
label2:
```

Compiled by Bibek Joshi

Commented [BJ1]:

```
printf("Largest value=%d",a);
label2:
printf("Largest value=%d",b);
}
```

LAB SESSION

For Loop Programs

1. A program to find the sum and the average of given numbers.

```
#include <stdio.h>
void main()
{
    int n, i;
    float sum=0.0, a, average;
    printf("enter the amount of number we want to add");
    scanf("%d",&n);
    for(i=1; i<=n; i++)
    {
        printf("Enter a number to be added:");
        scanf("%f",&a);
        sum=sum+a;
    }
    average=sum/n;
    printf("Sum=%f ", sum);
    printf("\nAverage= %f ", average);
}
```

2. A program to display the Fibonacci series.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int fibo3,fibo1=0,fibo2=1,num,i;
    clrscr();
    printf("enter the number of Fibonacci series to be displayed");
    scanf("%d",&num);
    printf("%d\t%d",fibo1,fibo2);
    for(i=3; i<=num; i++)
    {
        fibo3=fibo1+fibo2;
        fibo1=fibo2;
        fibo2=fibo3;
        printf("\t%d",fibo3);
    }
}
```

Compiled by Bibek Joshi

```
getch();  
}
```

3. If two numbers n1 & n2 are input through the keyboard. WAP to find sum of those numbers which is exactly divisible by 5 between n1 and n2.

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
int n1, n2, sum=0, i;  
printf("enter n1 & n2");  
scanf("%d%d",&n1, &n2);  
for(i=n1; i<=n2; i++)  
{  
if(i%5==0)  
sum=sum+i;  
}  
printf("required sum=%d",sum);  
getch();  
}
```

While Loop Program

4. Write a program to check whether the given number is prime number or not.

```
#include <stdio.h>  
#include <conio.h>  
void main()  
{  
int num, i;  
printf("Enter the number:");  
scanf("%d",&num);  
i=2;  
while(i<=num-1)  
{  
if (num%i == 0)  
{  
printf("%d is not a prime number",num);  
break;  
}  
i++;  
}  
if (i == num)  
printf("%d is prime number",num);  
getch();  
}
```

Compiled by Bibek Joshi

5. WAP to check whether it is palindrome or not. (Hint: A number is palindrome if it is equal to its reverse. For example: 4224 is palindrome but 4223 is not palindrome. To check whether a number is palindrome or not, first we should find the reverse of a number and check whether it is equal to its reverse or not.)

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,rev=0,y,d;
    printf("enter any number:");
    scanf("%d",&n);
    y=n;
    while(n!=0)
    {
        d=n%10;
        rev=rev*10+d;
        n=n/10;
    }
    if(rev==y)
        printf("%d is palindrome",y);
    else
        printf("%d is not a palindrome",y);
}
```

6. WAP to check whether a number is Armstrong number or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,m,sum=0,digit;
    printf("enter any number");
    scanf("%d",&n);
    m=n;
    while(n!=0)
    {
        digit=n%10;
        sum=sum+digit*digit*digit;
        n=n/10;
    }
    if(m==sum)
    {
        printf("%d is armstrong number",m);
    }
    else
    {
        printf("%d is not armstrong number",m);
    }
}
```

Compiled by Bibek Joshi

```

}
getch();
}

```

7. WAP to print all ASCII values & their equivalent characters using a while loop.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int i=0;
while(i<=255)
{
printf("%d %c\t",i,i);
i++;
}
getch();
clrscr();
}

```

Nested Loop Program

8. WAP to print the following output:

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

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

```

9. WAP to print the following output:

```

1 2 3 4 5
1 2 3 4

```

1 2 3
1 2
1

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

10. WAP to print the following output:

1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25

```
#include<stdio.h>
#include<conio.h>
void main() {
    int i, j, m;
    for(i=1; i<=5; i++) {
        for(j=1; j<=5; j++)
        {
            m=i*j;
            printf("%d\t", m);
        }
        printf("\n");
    }
}
```

11. WAP to print the following output:

1
2 3
4 5 6
7 8 9 10

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

Compiled by Bibek Joshi

```

void main()
{
int i, j, k=1;
for(i=1; i<=4; i++)
{
for(j=1; j<=i; j++)
{
printf("%d\t",k);
k++;
}
printf("\n");
}
getch();
clrscr();
}

```

12. WAP to print the following output

```

*
* *
* * *
* * * *

```

I method:

```

#include <stdio.h>
#include <conio.h>
void main()
{
int i, j, k;
for(i=1; i<=4; i++)
{
for(k=3; k>=i; k--) //this loop is used to provide space
{
printf (" ");
}
for(j=1; j<=i; j++)
{
printf ("%d\t",k);
}
printf("\n");
}
getch();
}

```

II Method:

Compiled by Bibek Joshi

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,k;
    for(i=4;i>=1;i--)
    {
        for(k=1;k<=i-1;k++)
        {
            printf("\t");
        }
        for(j=4;j>=i;j--)
        {
            printf("*\t");
        }
        printf("\n");
    }
    getch();
}

```

13. WAP to print the following output

```

* * * *
* * *
* *
*

```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j;
    for(i=1; i<=4; i++)
    {
        for(j=4; j>=i; j--)
        {
            printf("*\t");
        }
        printf("\n");
    }
    getch();
}

```