Programming for Problem Solving KCS 101

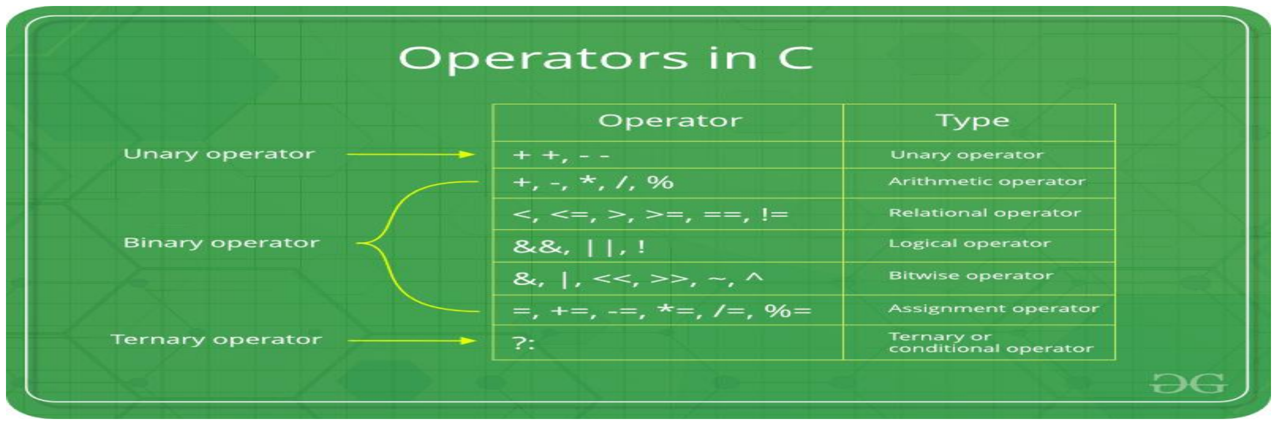
Unit 2

Operators , Bit Wise Operators ,Operator Precedence & Associativity,Type Casting & Type Conversion Condition Statements (if ,if else,switch)

Q Different types of Operators used in C Language and difference between relational and logical operator.

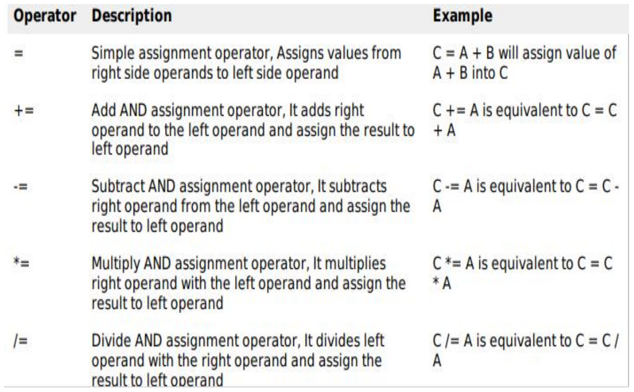
operators are the symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands.

For example, consider the below statement:
c = a + b; Here, '+' is the operator known as *addition operator* and 'a' and 'b' are operands. The addition operator tells the compiler to add both of the operands 'a' and 'b'.



Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, | |, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

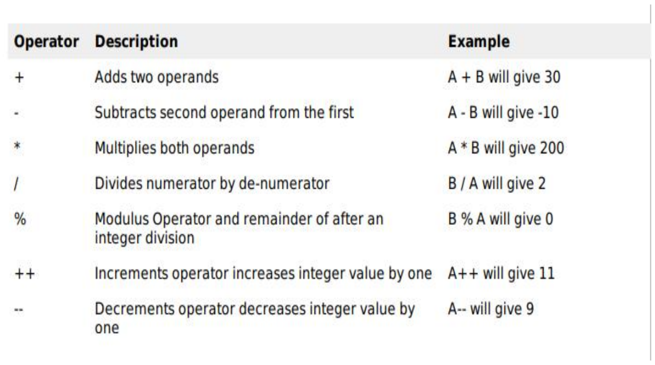
Assignment Operators : Assign the value



Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A

Arithmetic Operators

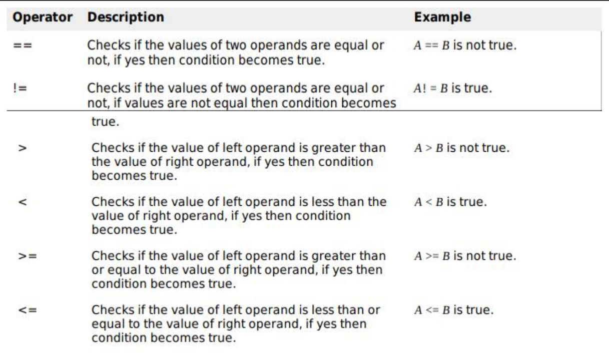
Perform all the Arithmetical operations. Table shows all the arithmetic operators supported by C language. Assume variable A holds 10 and variable B holds 20 then:



Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiplies both operands	A * B will give 200
/	Divides numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increments operator increases integer value by one	A++ will give 11
--	Decrements operator decreases integer value by one	A-- will give 9

Relational Operators

used to compare the value and it always display result in the form of 0 and 1 . if condition true its output is 1 and if false its output is 0.
Table shows all the relational operators supported by C language. Assume variable A holds 10 and variable B holds 20, then



Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	A == B is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	A! = B is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	A > B is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	A < B is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	A >= B is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	A <= B is true.

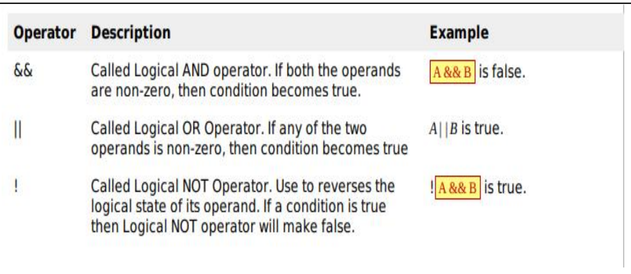
```
#include <stdio.h> void main()  {
 int x = 10; int y = 28;
if(x==y)
printf("Both variables are equal\n");
 if(x>y)
printf("x is greater than y \n");
if(x<y)
printf("x is less than y \n");
if(x!=y)
printf("x is not equal to y \n");
 if(x<=y)
 printf("x is lesser or equal to y\n");
if(x>=y)
printf("x is greater or equal to y \n");  }
```

Output

```
x is less than y    x is not equal to y    x is lesser or
equal to y
```

Logical Operators : compare between more than one conditions. Logical operators are used to perform logical operations. It returns 0 or 1 based on the result of condition, whether its true or false. These operators are used for decision making in C language.

Following table shows all the logical operators supported by C language. Assume variable A holds 1 and variable B holds 0, then



Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	A&&B is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true	A||B is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!A&&B is true.

```
#include <stdio.h>
void main() {
 int x = 10; int y = 28; int a = 15; int b = 20;
if(x<y && a==b)
printf("x is less than y AND a is equal to b\n");
if(x<y || a==b)
printf("x is less than y OR a is equal to b\n");
if(!x)
printf("x is zero\n");
 }
```

Output

```
x is less than y OR a is equal to b
```

Q Increment and decrement operator(unary)

Increment Operators are used to increased the value of the variable by one and Decrement Operators are used to decrease the value of the variable by one in C programs.

Both increment and decrement operator are used on a single operand or variable, so it is called as a unary operator. Unary operators are having higher priority than the other operators it means unary operators are executed before other operators.

```
++ // increment operator
-- // decrement operator
```

Type of Increment Operator

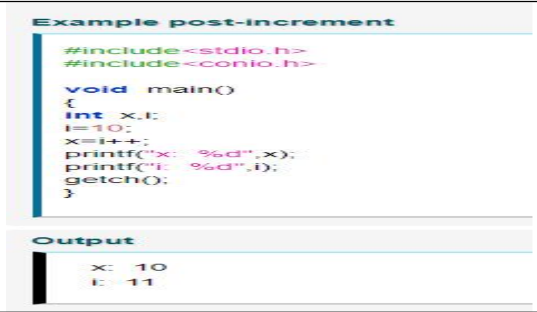
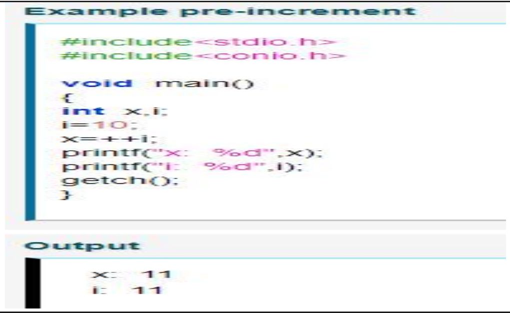
pre-increment
post-increment

pre-increment (++ variable)

In pre-increment first increment the value of variable and then used inside the expression (initialize into another variable).

post-increment (variable ++)

In post-increment first value of variable is used in the expression (initialize into another variable) and then increment the value of variable.

Example post-increment

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

void  main()
{
int  x,i;
i=10;
x=i++;
printf("x:  %d",x);
printf("i:  %d",i);
getch();
}
```

Output

```
x:  10
i:  11
```

Example pre-increment

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

void  main()
{
int  x,i;
i=10;
x=++i;
printf("x:  %d",x);
printf("i:  %d",i);
getch();
}
```

Output

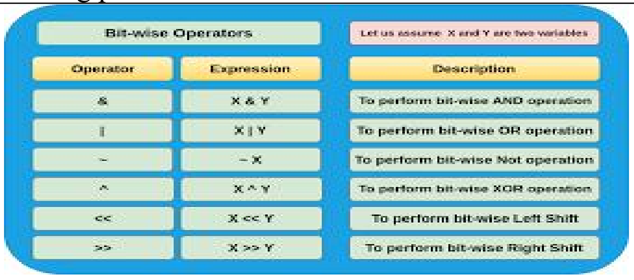
```
x:  11
i:  11
```

Q Bit Wise Operators

Bit Wise operator convert the data in to binary then perform the operation and again convert binary result into decimal from and give result to user.

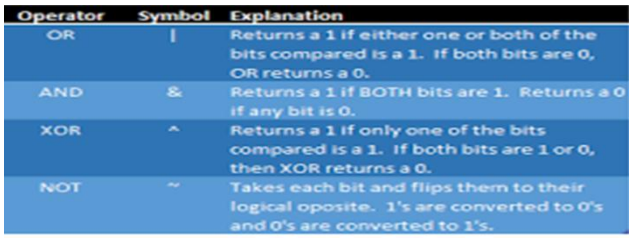
All the data stored in computer memory in the form of 0 and 1(bits form).to perform the operation on bits.

C provides six bitwise operators. These operator works with int and char type data. They cannot be used with floating point data.



Bit-wise Operators — Let us assume X and Y are two variables

Operator	Expression	Description
&	X & Y	To perform bit-wise AND operation
|	X | Y	To perform bit-wise OR operation
~	~ X	To perform bit-wise Not operation
^	X ^ Y	To perform bit-wise XOR operation
<<	X << Y	To perform bit-wise Left Shift
>>	X >> Y	To perform bit-wise Right Shift

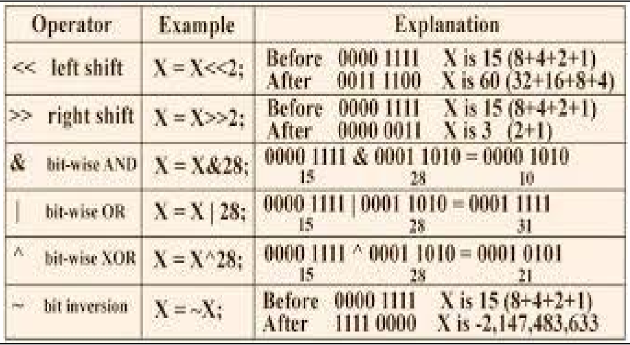


Operator	Symbol	Explanation
OR	|	Returns a 1 if either one or both of the bits compared is a 1. If both bits are 0, OR returns a 0.
AND	&	Returns a 1 if BOTH bits are 1. Returns a 0 if any bit is 0.
XOR	^	Returns a 1 if only one of the bits compared is a 1. If both bits are 1 or 0, then XOR returns a 0.
NOT	~	Takes each bit and flips them to their logical oposite. 1's are converted to 0's and 0's are converted to 1's.

```
#include <stdio.h>
Void  main() {
 int a = 60; /* 60 = 0011 1100 */
   int b = 13; /* 13 = 0000 1101 */
 int c = 0;
c = a & b; /* 12 = 0000 1100 *
/ printf("Line 1 - Value of c is %d\n", c );
c = a | b; /* 61 = 0011 1101 */
 printf("Line 2 - Value of c is %d\n", c );
 c = a ^ b; /* 49 = 0011 0001 */
 printf("Line 3 - Value of c is %d\n", c );
   c = ~a; /*-61 = 1100 0011 */
printf("Line 4 - Value of c is %d\n", c );
        c = a << 2; /* 240 = 1111 0000 */
   printf("Line 5 - Value of c is %d\n", c );
         c = a >> 2; /* 15 = 0000 1111 */
   printf("Line 6 - Value of c is %d\n", c ); }
```

When you compile and execute the above program, it produces the following result −

```
Line 1 - Value of c is 12
Line 2 - Value of c is 61
 Line 3 - Value of c is 49
```



Operator	Example	Explanation
<< left shift	X = X<<2;	Before 0000 1111 X is 15 (8+4+2+1) After 0011 1100 X is 60 (32+16+8+4)
>> right shift	X = X>>2;	Before 0000 1111 X is 15 (8+4+2+1) After 0000 0011 X is 3 (2+1)
& bit-wise AND	X = X&28;	0000 1111 & 0001 1010 = 0000 1010 15 28 10
| bit-wise OR	X = X | 28;	0000 1111 | 0001 1010 = 0001 1111 15 28 31
^ bit-wise XOR	X = X^28;	0000 1111 ^ 0001 1010 = 0001 0101 15 28 21
~ bit inversion	X = ~X;	Before 0000 1111 X is 15 (8+4+2+1) After 1111 0000 X is -2,147,483,633

Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15

Q Operator Precedence & Associativity

Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.

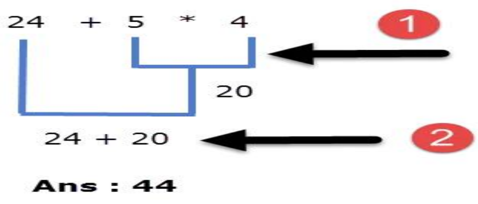
For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7
Like BODMAS RULE IN MATHS

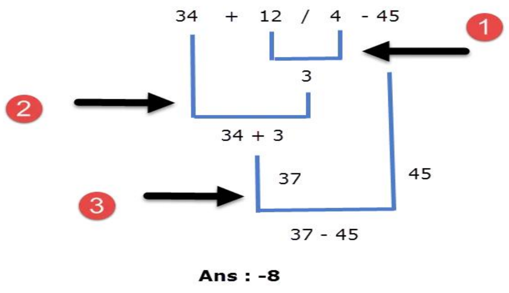
Associativity: It defines the order in which operators of the same precedence are evaluated in an expression. Associativity can be either from left to right or right to left.

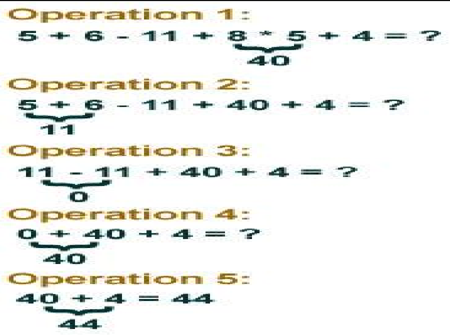
Precedence means when multiple operator of different sign comes in a single expression the order in which they are evaluated.

Associativity means when multiple operator of same sign comes in a single expression the order in which they are evaluated .

From the precedence table, we can conclude that the * operator is above the + operator, so the * operator has higher precedence than the + operator, therefore in the expression 24 + 5 * 4, sub expression 5 * 4 will be evaluated first then + evaluated.







Operation 1:
$5 + 6 - 11 + \underbrace{8 * 5}_{40} + 4 = ?$

Operation 2:
$\underbrace{5 + 6}_{11} - 11 + 40 + 4 = ?$

Operation 3:
$\underbrace{11 - 11}_{0} + 40 + 4 = ?$

Operation 4:
$\underbrace{0 + 40}_{40} + 4 = ?$

Operation 5:
$\underbrace{40 + 4}_{44} = 44$

OPERATOR	TYPE	ASSOCIAVITY
( ) [ ] . ->		left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

Q Type Casting & Type Conversion

In some cases we want/need to change the data type of variables. When we declare some variable as **int** the desired output may be float and vice versa. In such situations we may change the nature of data stored in a variable. This process is known as data Type conversion.

```
Void main()
{
int a = 10;
int b= 3;
int c= a/b;
printf("%d",c);
}
the output is  : 3
```

```
Void main()
{
int a = 10;
int b= 3;
float c= (float) a/b;
printf("%f",c);
}
the output is  : 3.3333
```

If we take c as float type then output is 3.00

In above Left example if we divide an integer value by integer the output should be integer, but if we want actual output we need type conversion. We can convert a data type into another data type like above Right Example.

Converting one datatype into another is known as type casting or, type-conversion. For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'. You can convert the values from one type to another explicitly using the **cast operator** as follows −
(type_name) expression

Consider the following example where the cast operator causes the division of one integer variable by another to be performed as a floating-point operation

```
#include <stdio.h>
void main()
 {
int sum = 17, count = 5;
double mean;
mean = (double) sum / count;
Printf("Value of mean : %f\n", mean );    }
```

When the above code is compiled and executed, it produces the following result −

```
Value of mean : 3.400000
```

It should be noted here that the cast operator has precedence over division, so the value of **sum** is first converted to type **double** and finally it gets divided by count yielding a double value.

Type conversions can be implicit which is performed by the compiler automatically, or it can be specified explicitly through the use of the **cast operator**. It is considered good programming practice to use the cast operator whenever type conversions are necessary

There are two types of typecasting.

1.Implicit Type casting − This conversion is done by the compiler. When more than one data type of variables are used in an expression, the compiler converts data types to avoid loss of data.
Here is an example of implicit type casting in C language,

```
#include <stdio.h>
void main() {
 int a = 10; char b = 'S'; float c = 2.88;
a = a+b;
printf("Implicit conversion from character to integer : %d\n",a);
c = c+a;
printf("Implicit conversion from integer to float : %f\n",c);     }
```

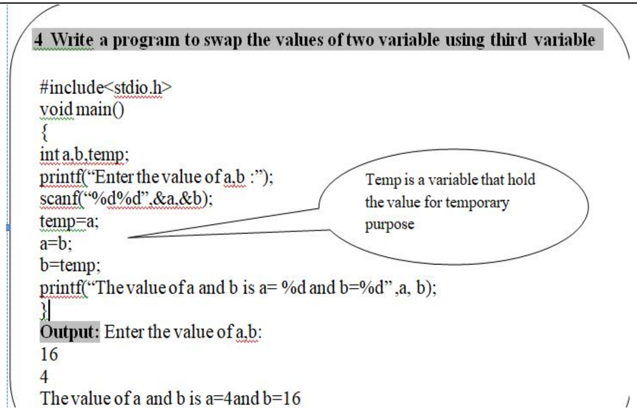
Output

```
Implicit conversion from character to integer : 93   Implicit conversion from integer to float : 95.879997
```

2.Explicit Type casting − This conversion is done by user. This is also known as typecasting. Data type is converted into another data type forcefully by the user.
Here is the syntax of explicit type casting in C language,
(type) expression Here is an example of explicit type casting in C language,

- #include <stdio.h>
- void main()
- {
- float c = 5.55;
- int s = (int)c+1;
- printf("Explicit Conversion : %d\n",s);   }
- **Output**
  Explicit Conversion : 6

C Programs

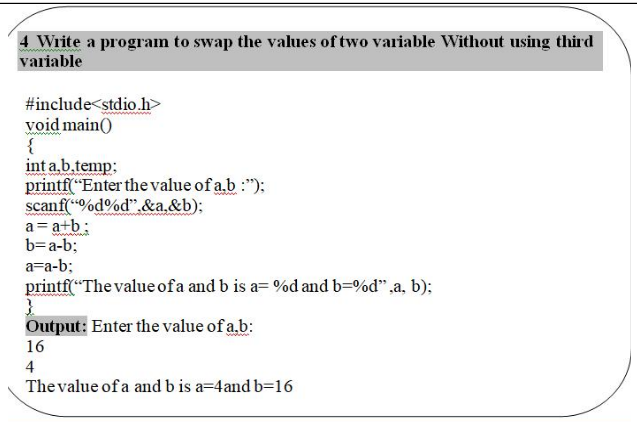


4 Write a program to swap the values of two variable using third variable

```
#include<stdio.h>
void main()
{
int a,b,temp;
printf("Enter the value of a,b :");
scanf("%d%d",&a,&b);
temp=a;
a=b;
b=temp;
printf("The value of a and b is a= %d and b=%d" ,a, b);
}
```

Temp is a variable that hold the value for temporary purpose

Output: Enter the value of a,b:
16
4
The value of a and b is a=4and b=16



4 Write a program to swap the values of two variable Without using third variable

```
#include<stdio.h>
void main()
{
int a,b,temp;
printf("Enter the value of a,b :");
scanf("%d%d",&a,&b);
a = a+b ;
b= a-b;
a=a-b;
printf("The value of a and b is a= %d and b=%d" ,a, b);
}
```

Output: Enter the value of a,b:
16
4
The value of a and b is a=4and b=16

- Programs

Write a program to find the Area of a circle.
Write a program to calculate the value in the following expressions.

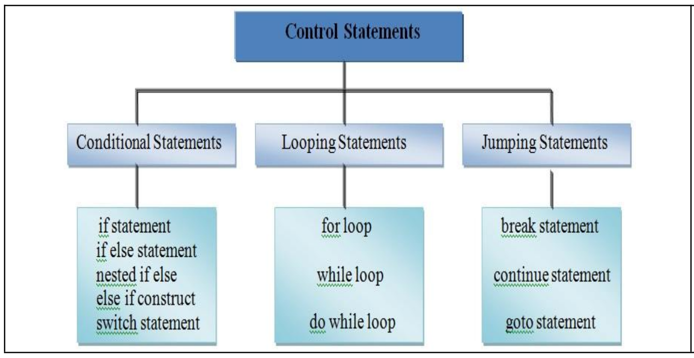
- r = a^5+3ab^2+b c) area = $\pi r^2+2\pi rh$
- area = $\sqrt{3}/4*a*a$ d) distance=ut+(at ^{2})/2

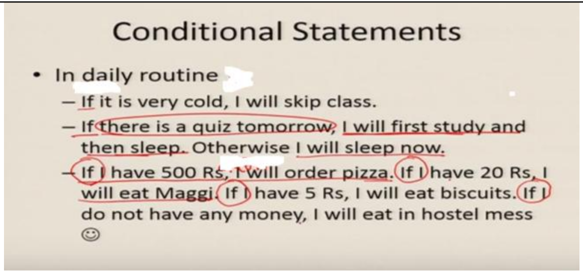
Write a program to take a number from user and find its square root.
Find the basic salary of an employee where DA is 40% of basic salary and hra is 20% of basic salary .and basic salary is entered by the user.

Write a program to find the sum of a 4 digit number.
Write a program to find the simple interest
Write a program to find the compound interest .
Write a program to take a 5 digit number and display the reverse of that number.
Give the difference between a) = and = = b) & and && c) a++ and ++a d) logical and relational operator
e) syntax and logical error F)local and global variable.

Find the value of following 97&35 , 125 | 79 , 35 ^67, 234<<3, 567>>3

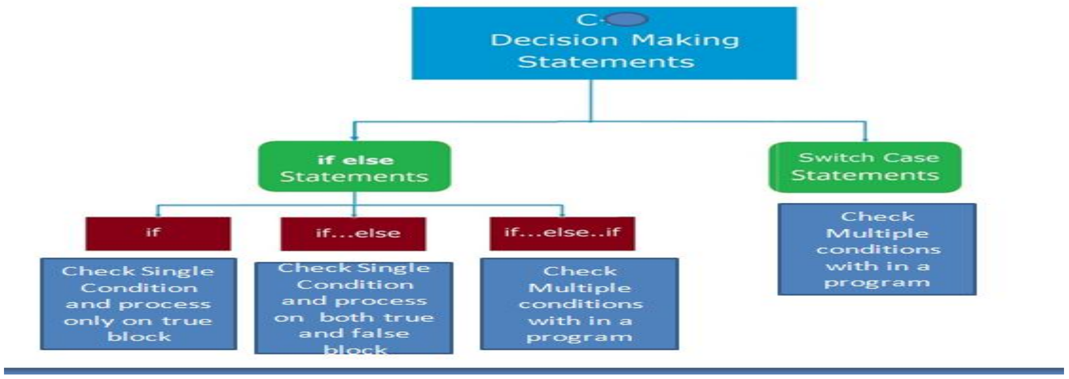
Condition Statements/Decision Making Statements/Branching




Conditional Statements

- In daily routine
  - If it is very cold, I will skip class.
  - If there is a quiz tomorrow, I will first study and then sleep. Otherwise I will sleep now.
  - If I have 500 Rs, I will order pizza. If I have 20 Rs, I will eat Maggi. If I have 5 Rs, I will eat biscuits. If I do not have any money, I will eat in hostel mess ☺

- Condition statements are those statements which provide the condition to execute a block. They are used to control the flow of execution of statements on the basis of some condition.
- This involve a kind of decision making capabilities to see whether a particular condition true or false
- if any condition is true then control is transfer to true block of statements and if false then control transfer to the false block of statements in a program.
- The conditional statement requires the programmer to specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.


C Decision Making Statements
if else Statements
Switch Case Statements
if
if...else
if...else..if
Check Single Condition and process only on true block
Check Single Condition and process on both true and false block
Check Multiple conditions with in a program
Check Multiple conditions with in a program

- If statement is used only for TRUE part of a single condition. Like if n = =5 then program display Hello. There is no any false part. If the condition true it display true block of statement otherwise control shift outside the true block.
- if else statement is used for both the TRUE part as well as FALSE Part of a single condition.
- Like if n= =5 then program display "hello" otherwise program display "bye" . There is a single statement for both the part if the condition is true then control transfer to the true block of statements and if false then control transfer to the false block of statements.

- Nested if else and switch both are used for when there is Multiple condition Means when more than one condition used in a program we use Nested if else and switch statements. like if n= =5 display “hello” and if n= =6 display “hi” and if n= =7 display “good morning” otherwise display “ bye” .so there are multiple conditions in a program so we use “Nested if else” or “Switch” statement

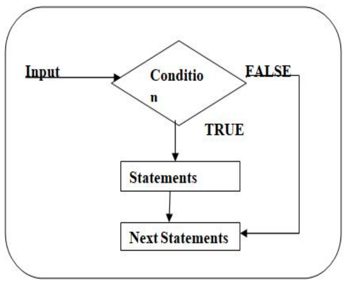
If Statement

if statement is used only for TRUE part of a single condition. It test the condition and the statement associated with is executed when the condition is true otherwise the statement are not executed at all.
it is also called one way branching.
If the result of the logical test is True(non zero value) then the statement is immediately follow if statement block and it is executed .
if the logical condition is false then control transfer to the next executable statement outside the body of if.

SYNTAX

```
if(condition)
{
Statement;
}
Next statement;
```

BLOCK DIAGRAM



Write a program to take the marks of a student and display fail if marks is less than 40%

```
void main()
{
int marks;
printf(“Enter the marks  (%) of a student”);
scanf(“%d”,&marks);
if  (marks < 40) {
printf(“The Student is FAIL”);
}}
```

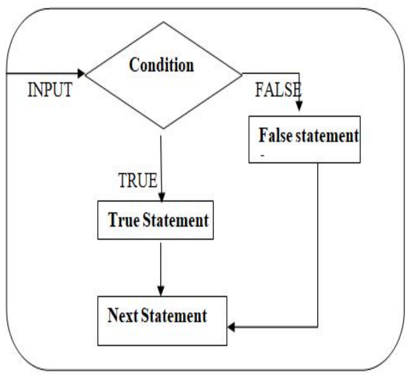
If else Statement

if else statement is used for TRUE part of a single condition as well as FALSE part also .the if else statement is two way branching.
There are two statements for a single condition.
If condition true statement 1 (that is defined inside the TRUE block) is executed .and when the condition is FALSE statement2 is executed that is defined in the else block. After this the program control is transfer on the next statement

SYNTAX

```
if (condition)
{
True Statement;
}
else
{
False Statement;
}
Next statement;
```

BLOCK DIAGRAM



Write a program to take the marks of a student and display fail if marks is less than 40% otherwise display The Student is PASS

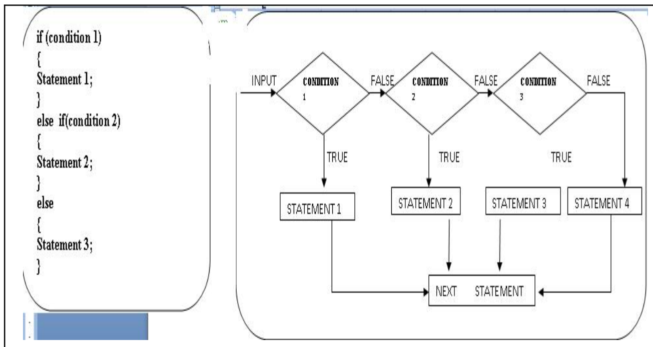
```
void main() {
int marks;
printf(“Enter the marks (%) of a student”);
scanf(“%d”,&marks);
if  (marks < 40){
printf(“The Student FAIL”);}
else{
printf(“The student PASS”);
}}
```

NESTED IF ELSE (if...else if Ladder)

Nested if else is used when there are multiple condition in a program .It is called multi way decision based on several conditions. Enclosing an if statement within another if, statement is called the nested if statement. When a series of decision are involved we may have To use more than one if,,.else statement in nested form .

The if...else statement executes two different codes depending upon whether the test expression is true or false. Sometimes, a choice has to be made from more than 2 possibilities.

The if...else if ladder allows you to check between multiple test expressions and execute different statements.



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

Write a program to take the marks of a student and display The Grades

```
#include<stdio.h>
void main()
{
int  marks;
printf(“Enter the marks(%) of a student”);
scanf(“%d”,&marks);
if  (marks >= 75){
printf(“The Student pass with A Grade”);  }
else  if (marks <  75 && marks  >=60) {
printf(“The student pass with B Grade”);    }
else if (marks <  60 && marks >=45)  {
printf(“The student is pass with C Grade”);   }
else{
printf(“The Student Fail”);
}   }
```

SWITCH STATEMENT

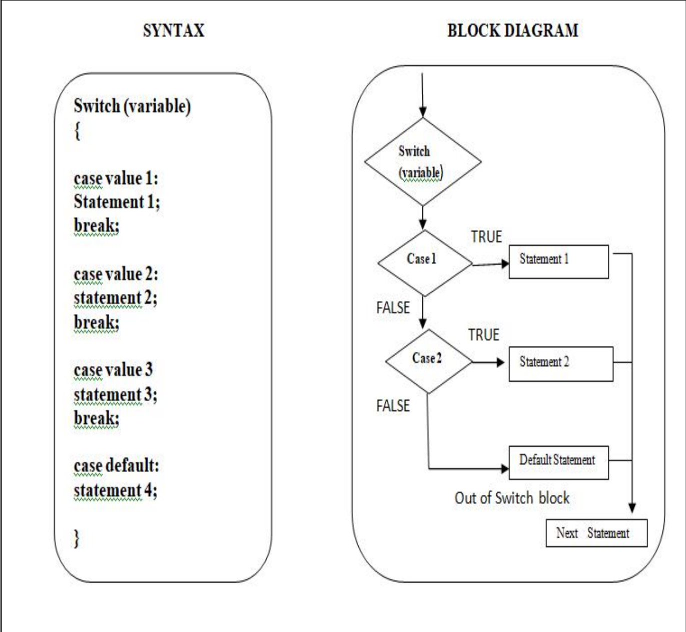
- The nested if else allow to selec
- t one of the many alternatives but it is time consuming .it test all the condition and based on the result a particular branch is taken for execution. To overcome this we used switch statement.
- The switch statement provide a multiple way of branching .it allow the user to select any one of the several alternatives, depending on the value of the expressions.

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**

The following rules apply to a **switch** statement −
The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.

The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.

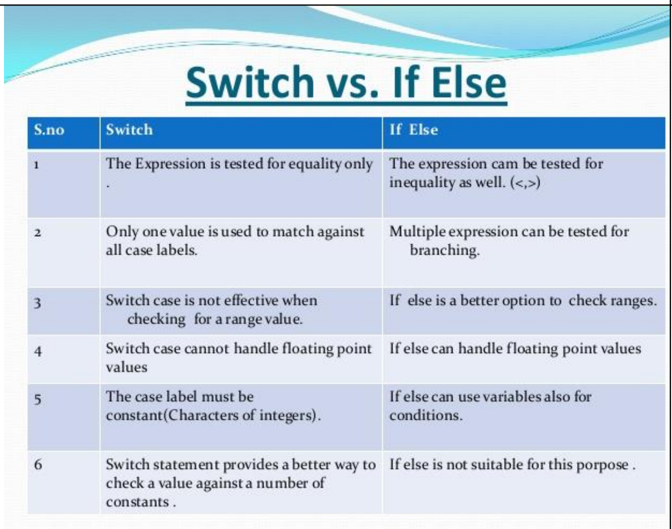
When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through* to subsequent cases until a break is reached.
A **switch** statement can have optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.



SYNTAX

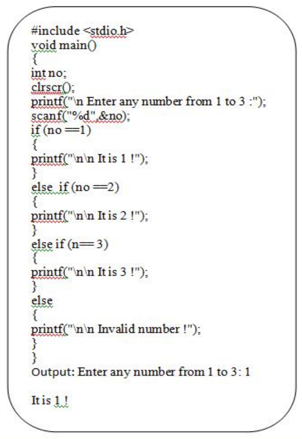
```
Switch (variable)
{

case value 1:
Statement 1;
break;

case value 2:
statement 2;
break;

case value 3
statement 3;
break;

case default:
statement 4;

}
```

```
#include <stdio.h>
void main ()
{
char grade ;
Printf("Enter your Grade");
Scanf("%c",&grade);
switch(grade) {
case 'A' :
printf("Excellent!\n" ); break;
case 'B' : case 'C' :
printf("Well done\n" ); break;
case 'D' : case 'E':
 printf("You passed\n" );break;
 case 'F' :
 printf("Better try again\n" ); break;
default :
 printf("Invalid grade\n" );
}}
```

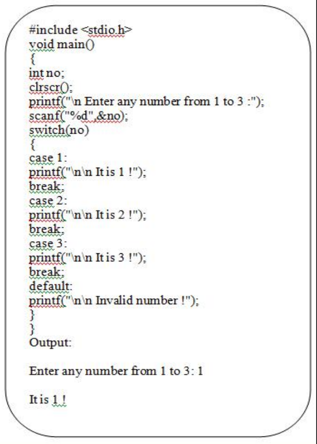


Switch vs. If Else

S.no	Switch	If Else
1	The Expression is tested for equality only .	The expression cam be tested for inequality as well. (<,>)
2	Only one value is used to match against all case labels.	Multiple expression can be tested for branching.
3	Switch case is not effective when checking for a range value.	If else is a better option to check ranges.
4	Switch case cannot handle floating point values	If else can handle floating point values
5	The case label must be constant(Characters of integers).	If else can use variables also for conditions.
6	Switch statement provides a better way to check a value against a number of constants .	If else is not suitable for this porpose .



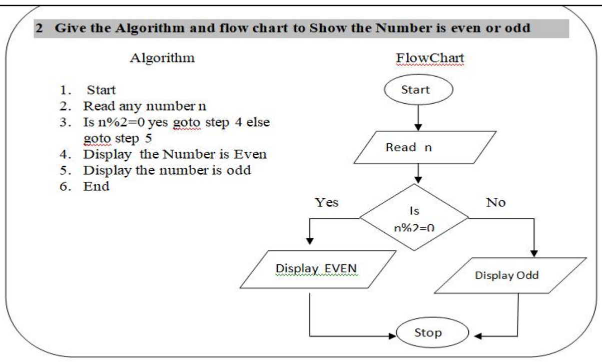
```
#include <stdio.h>
void main()
{
int no;
clrscr();
printf("\n Enter any number from 1 to 3 :");
scanf("%d",&no);
if (no ==1)
{
printf("\n\n It is 1 !");
}
else  if (no ==2)
{
printf("\n\n It is 2 !");
}
else if (n== 3)
{
printf("\n\n It is 3 !");
}
else
{
printf("\n\n Invalid number !");
}
}
Output: Enter any number from 1 to 3: 1

It is 1 !
```



```
#include <stdio.h>
void main()
{
int no;
clrscr();
printf("\n Enter any number from 1 to 3 :");
scanf("%d",&no);
switch(no)
{
case 1:
printf("\n\n It is 1 !");
break;
case 2:
printf("\n\n It is 2 !");
break;
case 3:
printf("\n\n It is 3 !");
break;
default:
printf("\n\n Invalid number !");
}
}
Output:

Enter any number from 1 to 3: 1

It is 1 !
```

WAP to display the Entered number is even or odd

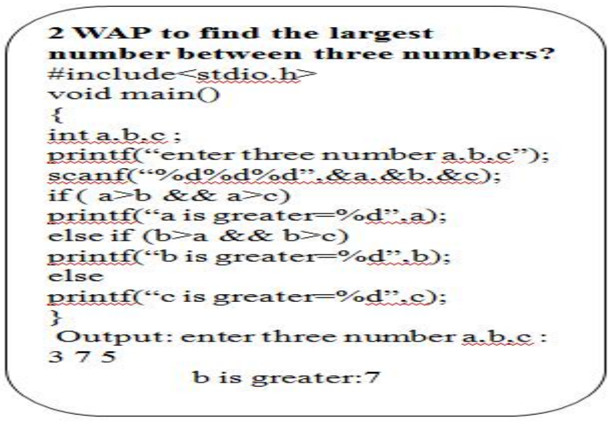
```
# include<stdio.h>
void main()
{
int  n;
printf("Enter any number:");
scanf("%d",&n);
if ( n%2 ==0)
printf("The number is Even");
else
printf("The number is odd"); }
```



2 Give the Algorithm and flow chart to Show the Number is even or odd

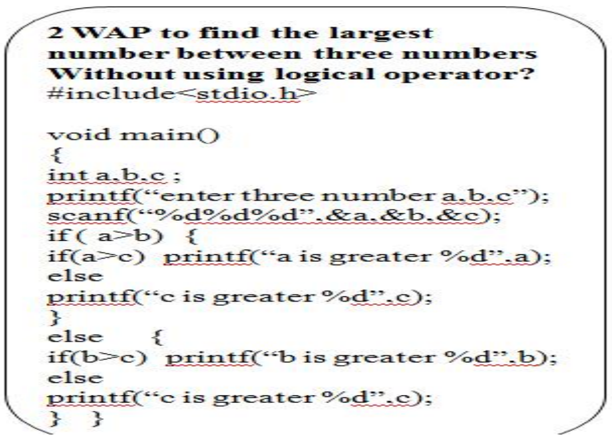
Algorithm

```
1. Start
2. Read any number n
3. Is n%2=0 yes goto step 4 else
   goto step 5
4. Display  the Number is Even
5. Display the number is odd
6. End
```



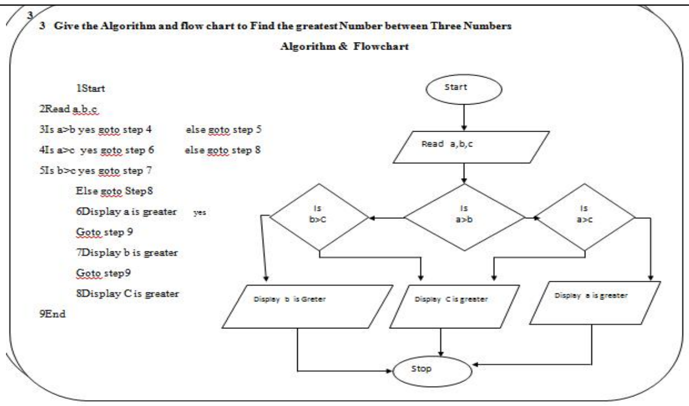
2 WAP to find the largest number between three numbers?

```
#include<stdio.h>
void main()
{
int a,b,c ;
printf("enter three number a,b,c");
scanf("%d%d%d",&a,&b,&c);
if ( a>b && a>c)
printf("a is greater=%d",a);
else if (b>a && b>c)
printf("b is greater=%d",b);
else
printf("c is greater=%d",c);
}
 Output: enter three number a,b,c :
3 7 5
          b is greater:7
```



2 WAP to find the largest number between three numbers Without using logical operator?

```
#include<stdio.h>

void main()
{
int a,b,c ;
printf("enter three number a,b,c");
scanf("%d%d%d",&a,&b,&c);
if ( a>b)  {
if(a>c)  printf("a is greater %d",a);
else
printf("c is greater %d",c);
}
else     {
if(b>c)  printf("b is greater %d",b);
else
printf("c is greater %d",c);
}   }
```



3 Give the Algorithm and flow chart to Find the greatest Number between Three Numbers

Algorithm & Flowchart

```
1Start
2Read a,b,c
3Is a>b yes goto step 4        else goto step 5
4Is a>c  yes goto step 6       else goto step 8
5Is b>c yes goto step 7
        Else goto Step8
        6Display a is greater    yes
        Goto step 9
        7Display b is greater
        Goto step9
        8Display C is greater
9End
```

C Program to Make a Simple Calculator Using switch...case

```
#include <stdio.h>
 void main()     {
char operator;
int first, second;
 printf("Enter an operator : ");
scanf("%c", &operator);
printf("Enter two operands: ");
 scanf("%d %d", &first, &second);
switch (operator)  {
case '+':  printf("%d", first + second);  break;
 case '-':  printf("%d", first - second); break;
case '*':  printf("%d, first * second); break;
case '/':  printf("%d, first / second); break;
default:  printf("Error! operator is not correct");
 } }
```

Conditional operator /Ternary Operator   ? :

- This conditional operator is also known as the Ternary Operator. This operator has three phase.
- **Exp1 ? Exp2 : Exp3;**
- where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon. The value of a ? expression is determined like this: Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire ? expression. If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the expression.
- The ? is called a ternary operator because it requires three operands and can be used to replace if-else statements.

For example, consider the following code

```
if(y < 10) {
var = 30; }
else {
var = 40; }
```

Above code can be rewritten like this

```
var = (y < 10) ? 30 : 40;
```

find greater number between Three Numbers Using Conditional operator/Ternary Operator

```
# include <stdio.h>
void main() {
int a, b, c, big ;
 printf("Enter three numbers : ") ;
scanf("%d %d %d", &a, &b, &c) ;
 big = a > b ? (a > c ? a : c) : (b > c ? b : c) ;
 printf("\nThe biggest number is : %d", big); }
```

1.WAP to number is positive or negative
2 WAP to check Even-Odd number
3. WAP find greater between two numbers
4. Program to find the average of marks obtained by student and display Division
5. WAP Arrange 3 numbers in increasing order.
6. An electronic power company charges its customer as follow:If unit is 0 to 100 then rent is 100 and rate of charge is 1 rs per unit If unit is 100 to 200 then rent is 150 and rate of charge is 1.50 rs per unit If unit is 200 to 300 then rent is 200 and rate of charge is 2 rs per unit For above 300 then rent is 250 and rate is 2.50 rs per unit. Print the amount to be paid by the customer.
7. Write a program to evaluates the function
$F(x) = ax^2+b$ if $b>=5$
$F(x) = ax + bx$ if $b<5$
8. Write a program to read an integer number from the keyword ,add 1 to it ,it the number is even and again add 1 to it if the number is less than 20 otherwise keep the number unchanged and display that number.
9. If the basic salary is less than Rs 1500 ,then HRA is 10% of the basic salary and DA is 90% of basic salary .if the salary is either equal to or above 1500 then HRA is 500 and DA is 95% of basic salary .if the employee salary is entered then find the gross salary and display it.
10. Write a program that allow the user to enter in 5 grads marks between 0-100 .the program must calculate the average marks ,and state the number of marks is less than 65.