* Types of operators
  - Arithmetic operator
  - Logical operator
  - Relational operator
  - conditional or Ternary operator
  - Bitwise operator
  - Increment & Decrement operator
  - Assignment operator
  - Unary operator
  - Special operator
* Type casting & Type conversion
* operator precedence & Associativity
* Evaluation of Expression

* Operator

An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.
Eg:- +, - etc.,

* operand

operand is a variable or constant which returns a value.
Eg:- a+b
a & b are variable
+ is a operator

* Expression

A ~~sequence~~ sequence of operands &

operators that reduces to single value.

*Classification of operators

1. Based on number of operands

a) Unary operators Eg:- +a, -10 etc.,
b) Binary operators Eg:- a+b
c) Ternary operators Eg:- ? and :
d) Special operators Eg:- comma.

2. Based on type of operation

a) Aithemetic operators
b) Relational operators
c) Logical operators
d) Assignment operators
e) Increment & Decrement operators
f) Conditional operator
g) Bitwise operator
h) Special operator

1) * Aithmetic operators

C operation	Aithmetic operator
Addition	+
subtraction	-
Multiplication	*
Division	/
Modulus	%

* Arithmetic operations

- Integer Arithmetic
- Real Arithmetic
- Mixed mode Arithmetic

• Integer Arithmetic

When both the operands are integer type in a single arithmetic expression its called integer expression & the operation is called integer arithmetic.

Eg:- int a=10, b=4;

a+b=14

a-b=6

a*b=40

a/b=2 (Decimal part truncated)

a%b=2 (remainder)

• Real arithmetic

An arithmetic expression involving only real operands.

Eg:- float x, y, z;

x = 6.0/7.0 = 0.857143

y = 1.0/3.0 = 0.333333

z = -2.0/3.0 = -0.666667

• Mixed mode arithmetic

When 1 of the operand is real & the other is integer. then expressi

is mixed mode.

Eg:- int a=15; float b=10.0;
a/b = 15/10.0 = 1.5
where as 15/10 = 1

1) Write a C program showing the usage of arithmetic operators.

```
#include<stdio.h>
void main()
{
  int a,b,c,d,e,f,g;
  clrscr();
  a=10;
  b=2;
  c=a+b;
  d=a-b;
  e=a*b;
  f=a/b;
  g=a%b;
  printf("%d+%d=%d\n",a,b,c);
  printf("%d-%d=%d\n",a,b,d);
  printf("%d*%d=%d\n",a,b,e);
  printf("%d/%d=%d\n",a,b,f);
  printf("%d%%%d=%d\n",a,b,g);
  getch();
}
```


Output :-
c = 12 → 10+2 = 12
d = 8 → 10-2 = 8
e = 20 → 10*2 = 20
f = 5 → 10/2 = 5
g = 0 → 10%2 = 0

2) * Relational operator

It is a operator used to specify the relationship between 2 operands.

operator	Meaning
<	Less than
>	Greater than
<=	Less than/equal to
>=	Greater than/equal to
==	checks values are equal
!=	Not equal

* If the condition is true then prints 1
* If the condition is false then prints 0.

2) Write a C program showing the usage of Relational operators.

```
#include<stdio.h>
void main()
{
  int a, b;
  clrscr();
  a=10;
  b=20;
printf("%d<%d=%d", a,b, a<b);
printf("%d>%d=%d", a,b, a>b);
printf("%d<=%d=%d", a,b, a<=b);
printf("%d>=%d=%d", a,b, a>=b);
printf("%d==%d=%d", a,b, a==b);
printf("%d!=%d=%d", a,b, a!=b);
getch();
}
```

output

10<20=1
10>20=0
10<=20=1
10>=20=0
10==20=0
10!=20=1

3) * Logical operators

It is a operator used to perform logical operations on the given expression.
An expression containing logical operator returns 0 or 1 depending upon whether expression results true or false.

operator	Meaning
&&	- Logical AND. True only if all operands are true
!	- Logical NOT. True only if the operand is 0.
||	- Logical OR. True only if either one operand is true.

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a, b, c, d;
  clrscr();
```

```
a=10; b=11; c=12;
d=(a<b) && (c>b);
printf(" d value is %d", d);
d=(a>b) || (c>b);
printf(" d value is %d", d);
d=!(a!=b);
printf(" d value is %d", d);
getch();
}
```


output

```
d value is 1
d value is 1
d value is 0
```

4) * Assignment operator

* Assignment operator is a operator used to assign value to the variable
* the most common assignment operator is =.
  - Eg:- int a;
    a=10; // assignment

operator	Example (Short hand operator)	same as
=	a=b	a=b
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

Example

```
#include<stdio.h>
void main()
{
    int a=5, b;
    b=a;
    printf("b=%d\n", b);
    b+=a;
    printf("b=%d\n", b);
    b-=a;
    printf("b=%d\n", b);
    b*=a;
    printf("b=%d\n", b);
    b/=a;
    printf("b=%d\n", b);
    b%=a;
    printf("b=%d\n", b);
    getch();
}
```

output

```
b = 5
b = 10
b = 5
b = 25
b = 5
b = 0
```

* Multiple assignment operator → used to assign a common value to 2 or more variable.

Eg:- int a=b=c=1; // multiple assignment.

5) * Increment & decrement operator

* Increment operator used to increase the value of variable by one.

* Decrement operator used to decrease the value of variable by one.

* There are 2 types of increment operator.

* pre-increment :- It increments the value before assigning to the variable
Eg:- ++i;

* post-increment :- It increments the value after assigning to the variable.
Eg:- i++;

* There are 2 types of decrement operator.

* pre-decrement :- It decrements the value before assigning to a variable
Eg :- --i;

* post-decrement :- It decrements value after assigning to a variable
Eg:- i--;

Example 1:-

```
#include <stdio.h>
void main()
{
  int a, b, c, d;
  clrscr();
  a = 10;
  b = 15;
  c = 20;
  d = 25;
  printf("++a = %d\n", ++a);
  printf("++b = %d\n", ++b);
  printf("--c = %d\n", --c);
  printf("--d = %d\n", --d);
  getch();
}
```


output

```
++a = 11
++b = 16
--c = 19
--d = 24
```

Example 2:-

```
int a=2, b=10, c=0, p, q, r;
p = a++ + b-- + ++c;
q = --a + b++ + ++c;
r = a-- + --b + c++;
```

output

a=1, b=9, c=3, p=13, q=13, r=13

6) * Conditional operator

* It is also called as ternary operator.
* It is used to test relationship between 2 variables.
* It takes 3 operands.
* The syntax is:-

<expression> ? <value 1> : <value 2>;

where,
'?' - used as a conditional operator.
'expression' - Relational expression
'value 1' - value to be assigned when the result of expression is 'true'.
'value 2' - value to be assigned when the result of expression is 'false'.

Eg:- c = a > b ? a : b;

Here, c will be assigned the value of a if a is greater than b. Otherwise, c will be assigned the value of b.

Example :-

```
#include<stdio.h>
void main()
{
  int a=5, b=4, c;
  clrscr();
```

```
    c = a>b ? a : b;
    printf("c = %d\n", c);
    getch();
}
```

output

c = 5

7) * Bitwise operator :-

*Bitwise operator is a operator used to perform bit-level operations

operators	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise complement
<<	Left Shift
>>	Right Shift

a) Bitwise AND operator :-

* the output of bitwise AND is 1 if the corresponding bits of 2 operands is 1.
* If either bit of an operand is 0, the result is evaluated to 0.

- 0 & 0 = 0
- 0 & 1 = 0
- 1 & 0 = 0
- 1 & 1 = 1

Eg:- 12 & 25

```
  0000 0000 0000 1100 (Binary 12)
& 0000 0000 0001 1001 (Binary 25)
-----------------------
  0000 0000 0000 1000 → 8
```

12 & 25 = 8

b) Bitwise OR operator

* The output of bitwise OR is 1 if atleast one corresponding bit of 2 operands is 1.

0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

Eg:- 12 | 25

```
  0000 0000 0000 1100
| 0000 0000 0001 1001
-----------------------
  0000 0000 0001 1101 → 29
```

12 & 25 = 29

c) Bitwise XOR operator

If the corresponding bits of 2 operands are different then the result is 1.

0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

Eg:- 12 ^ 25

```
0000 0000 0000 1100
0000 0000 0001 1001
---------------------
0000 0000 0001 0101  = 21
```

12 ^ 25 = 21

d) Bitwise complement operator

Unary operator which changes 1 to 0 and 0 to 1.

$$\boxed{\sim n = -(n+1)}$$

$\sim 0 = 1$

$\sim 1 = 0$

Eg:- $\sim 12 = -13$

```
        0000 0000 0000 1100
~   ---------------------
        1111 1111 1111 0011  -> -13
```

For negative numbers

$n = -12 \rightarrow \sim -12 = ?$

$\sim n = -(n+1)$

$\sim n = -(-12+1)$

$\sim -12 = -(-11)$

$\sim -12 = 11$

Represent 12 in binary & find 2's complement i.e., nothing but -12.

```
0000 0000 0000 1100
---------------------
1111 1111 1111 0011  -> 1's
                   1 -> 2's Complement
---------------------
1111 1111 1111 0100  -> -12
```

~-12 = 0000 0000 0000 1011 → 11

e) Bitwise left shift operator

Left shift operator shifts all bits towards left by certain number of specified bits.

12 = 0000 0000 0000 1100

12<<2 =

0000 0000 0000 1100

0000 0000 0011 0000 → 48

∴ 12<<2 = 48

a<<b = a * 2^b

i.e., 12<<2 = 12 * 2^2

= 12 * 4

= 48

2 | 11
2 | 5 - 1
2 | 2 - 1
1 - 0

11 → 1011

f) Bitwise right shift operator

It shifts all bits towards right by certain number of specified bits.

12 = 0000 0000 0000 1100

12>>2 = 0000 0000 0000 1100

0000 0000 0000 0011 → 3

∴ 12>>2 = 3

a>>b = a / 2^b

12>>2 = 12 / 2^2 = 12/4 = 3

Example

```
#include<stdio.h>
void main()
{
    int a=60, b=13;
    int c=0;
    c= a & b;
    printf(" c= %d\n", c);
    c= a | b;
    printf(" c= %d\n", c);
    c= a ^ b;
    printf(" c= %d\n", c);
    c= ~a;
    printf(" c= %d\n", c);
    c= a<<2;
    printf(" c= %d\n", c);
    c= a>>2;
    printf(" c= %d\n", c);
    getch();
}
```


8-bits

60 = 00111100

13 = 00001101

Output

```
c is 12
c is 61
c is 49
c is -61
c is 240
c is 15
```

9) special operators

1. comma operator
2. sizeof() operator
3. Address operator
4. Dot operator

1. Comma operator

* It is used to link the related expressi together.

Eg:- int a, b, c=5;

2. sizeof() operator

It is a unary operator which returns the size of an operand.

Eg:-

```
#include<stdio.h>
void main()
{
clrscr();
printf("%d\n", sizeof(char));
printf("%d\n", sizeof(int));
printf("%d\n", sizeof(float));
printf("%d\n", sizeof(double));
getch();
}
```

1) b) Design and develop a C program to find the largest of three numbers using Ternary operator.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a, b, c, big;
  clrscr();
  printf("Enter the value of a, b and c\n");
  scanf("%d%d%d", &a, &b, &c);
  big = (a>b && a>c)? a : (b>c)? b : c;
  printf("Largest number is %d", big);
  getch();
}
```

Output :-

Enter the value of a, b and c
3
7
4
Largest number is 7.

CHAITHRA. H.E.
B.E, M. Tech

Write a program to find whether a given number is even or odd using ternary operator.

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

operators precedence & Associativity

operator	priority	Associativity
(), [], .	1	L to R
++, --, !, ~	2	R to L
*, /, %	3	L to R
+, -	4	L to R
<<, >>	5	L to R
<, <=, >, >=	6	L to R
==, !=	7	L to R
&, ^, |	8	L to R
&&, ||	9 & 10	L to R
?:	11	L to R
=, +=, -=, *=, /=, %=	12	R to L
,	13	L to R

Eg:- 1) $10 - 3 \% 8 + 6 / 4$

$10 - 3 + 6 / 4$

$10 - 3 + 1$

$7 + 1$

$= 8$

2) $17 - 8 / 4 * 2 + 3 - ++a$

$a = 5$

$17 - 8 / 4 * 2 + 3 - 6$

$17 - 2 * 2 + 3 - 6$

$17 - 4 + 3 - 6$

Date
Page

$13 + 3 - 6$

$16 - 6$

$= 10$

3) $5 + ((6-3) * (5-1))$

$5 + (3 * (5-1))$

$5 + (3 * 4)$

$5 + 12$

$= 17$

CHAITHRA. H.E.
B.E, M. Tech.

* Type casting & Type conversion

*Type conversion occures when the expression has data of mixed data types.

* It is the process of converting one type of data to another type.

* It is of 2 types

1) Implicit Type conversion (Type promotion)
2) Explicit Type conversion (Type casting)

1) Implicit type conversion :-

*when the type conversion is performed automatically by the compiler without programmers intervension, such type conversion is called implicit type conversion.

1)
```
#include<stdio.h>
void main()
{
    float b;
    b=150;
    printf("b=%f\n",b);
}
```

Output

b=150.000000

* In type conversion, the data type is promoted from lower to higher because converting higher to lower leads to loss of precision & value.

2) Explicit Type conversion:-

* It can be applied on any expression with a unary operator called a cast.

Syntax:-

(type-name) expression;

Eg:-
```
int n;
float x;
x = (float) n;
```

* In the above statement, it will convert the value of n to a float value before assigning to x. But n value is not altered.

* Type casting doesnot change the actual value of variable, but the resultant value may be changed & stored.

Example:-

```
#include<stdio.h>
Void main()
{
    int a,b;
    float c,d;
    clrscr();
```

```
a = 1;
c = 3.5;
b = (int) c;  ⟶ b = 3
d = (float) a / (float) b;  ⟶ d = 1.0 / 3.0
printf("d = %f", d);  ⟶ d = 0.333
getch();
}
```

output

d = 0.333