Unit:- 2

Fundamentals Of C

⇒ Points:-

- Features of C Language
- Structure of C Program
- Comments
- Header files
- Data-types
- Constants & Variables
- Operators
- Expressions
- Evaluation of Expressions
- Type Conversion
- Precedence & associativity
- I/O functions

Prepared By:-

Ms. Jinal Zala
(M.E (CE), Asst Prof,
LJIET

→ Features of C language :-

1. It is a robust language with rich set of built-in functions & operators that can be used to write any complex program.

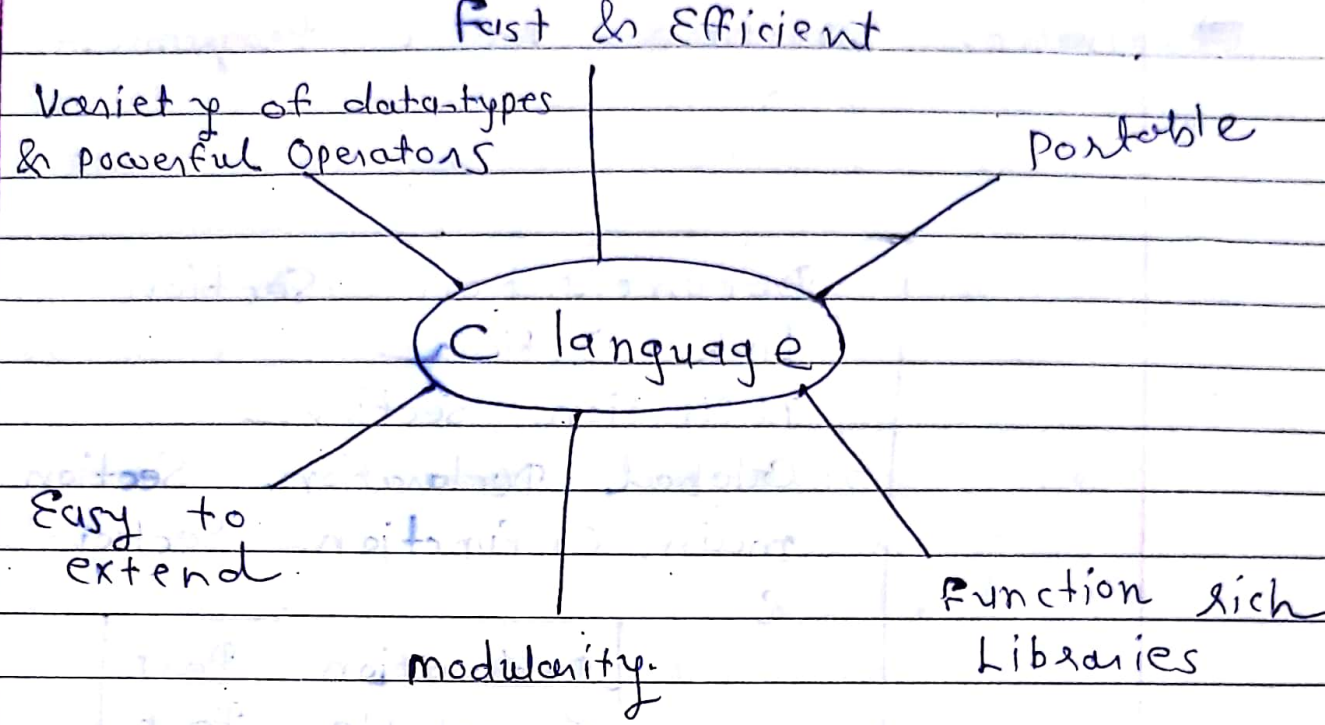
2. Programs written in C are efficient & fast. This is due to its variety of data-type & powerful operators.

3. It is many time faster than BASIC.

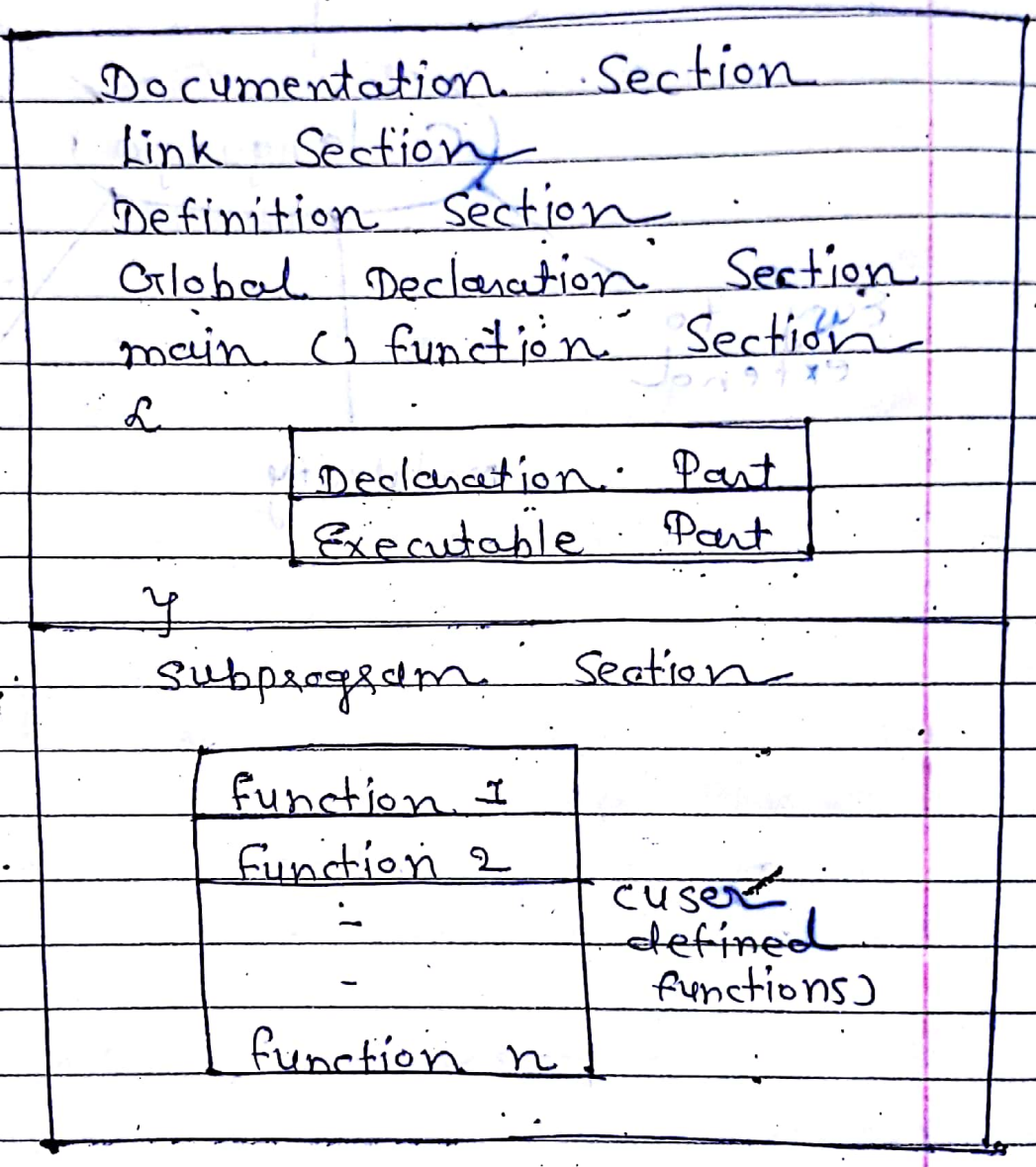
4. C is highly portable this means that programs once written can be run on another machine with little or no modification.

5. A C program is basically a collection of functions that are supported by c library. We can also create our own function & add it to C library.

6. Another important feature of C is its ability to extend itself. (We can continuously add our own functions to C library.)



↦ Basic Structure of C Programs :-



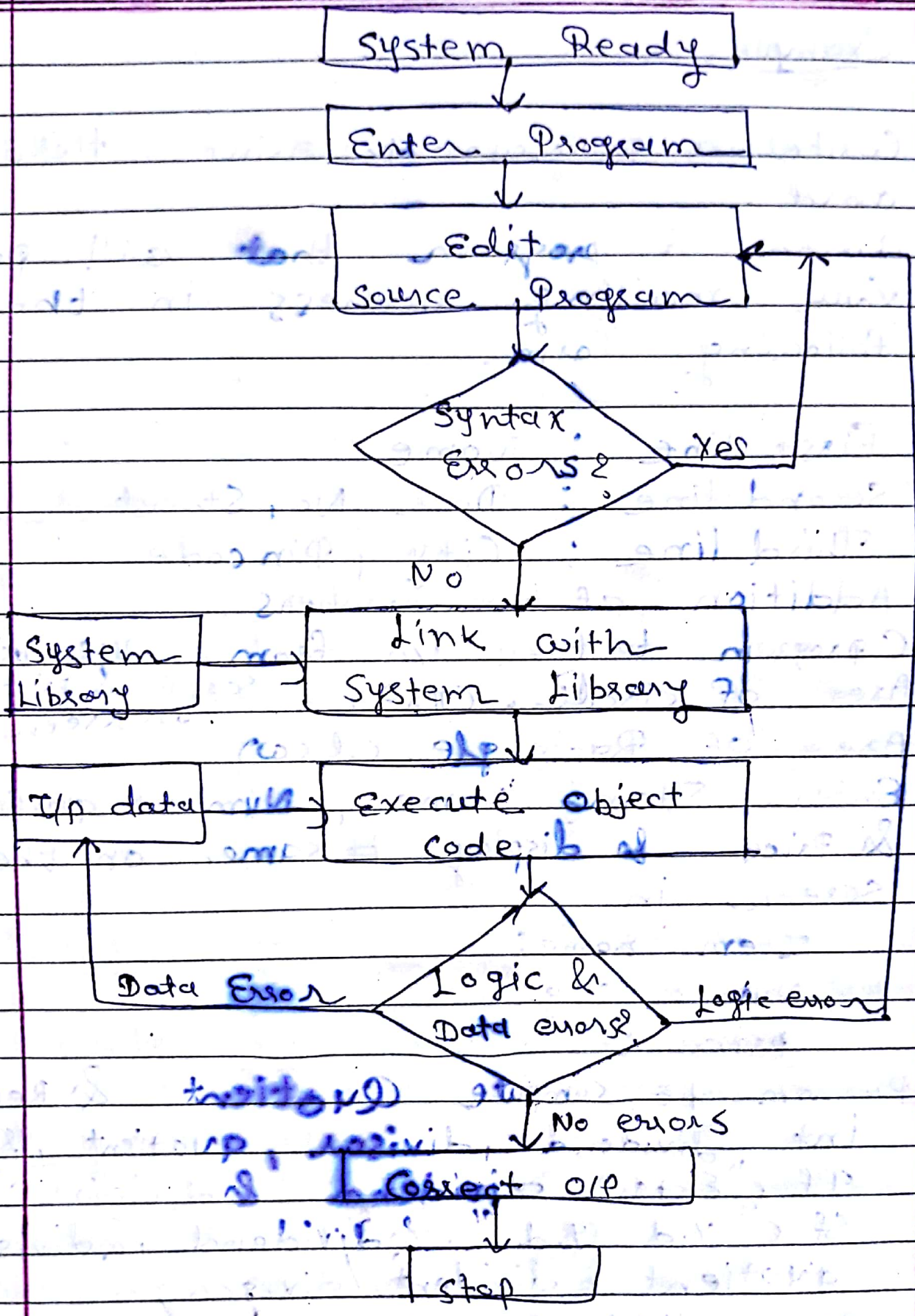
↦ The documentation section consists of a set of comment lines giving the name of the program, the author & other details.

Date
Page

- The link section provides instructions to the compiler to link functions from the system library.
- The definition section defines all symbolic constants.
- There are some variables that are used in more than one function. Such variables are called global variables & are declared in the global declaration section.
- Every C program must have one main() function section.
  - This section contains two parts
    - Declaration Part
    - Executable Part
  - The declaration Part declares all the variables used in the executable part.
  - All the statements in main sections are ending with a semicolon.

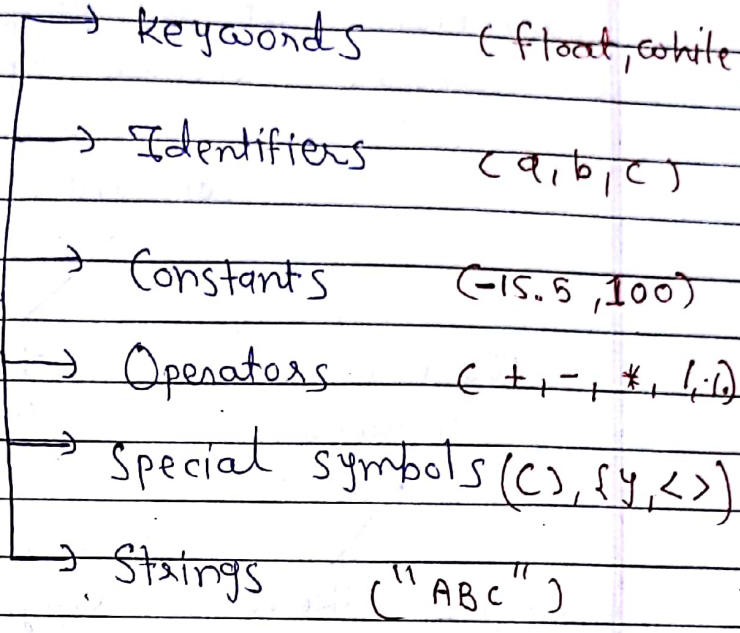
⇒ Executing a 'C' Program :-

1. Creating the Program
2. Compiling the Program
3. Linking the Program with functions that are needed from the C library
4. Executing the program



★ C Tokens:-

↦ Smallest individual units in C Programming are knoans as 'C Tokens'.



↦ Keywords:-

- All keywords have fixed meanings & these meaning can't be changed.
- Predefined words
- There are 32 keywords in C.

→ Ex:- float, int, break, case

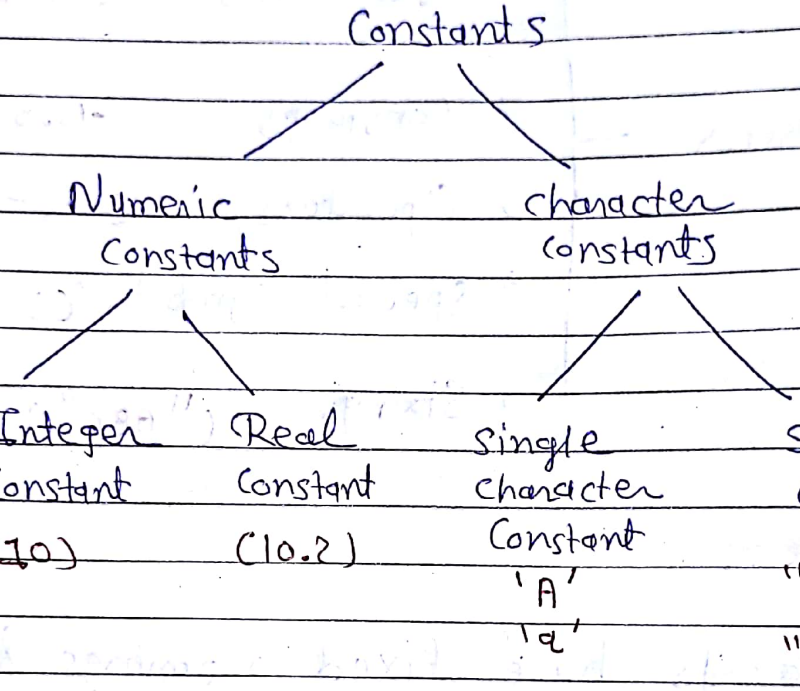
↦ Identifies:-

- identifiers are nothing but variable's name.
- Rules:-

1. First character must be an alphabet.
2. Must not contain white space.
3. Can't use a keyword.
4. Must consist of only letters, digits or underscore.

↦ Constants :-

- Constants in C refer to fixed values that do not change during the execution of a program.
- C supports several types :-



Ex :- (10) (10.2) 'A' 'a' "ABC" "xyz"

↦ Operators :-

- C provides basic arithmetic operators +, -, /, *, %.

↦ Special symbols :-

- Special symbols have some special meaning in the C Programming language for the compiler & it is used to perform special task in C.

Ex:- d y , ( ) , < > , ; , : , ,

↦ Strings :-

- Sequence of characters are known as 'strings'.
- String always represent in "double quotes"

Ex:- "ABC" , "PPS".

★ Types of Data-types :-

1. Primary data-types
2. Derived data-types
3. User defined data-types,

↦ Primary data-types :-

- Predefined or builtin datatypes are known as Primary - data types.
- All C compilers support five primary data-types.

Data-type	Size	Range
char	1 byte	-128 to 127
int	2 bytes	-32,768 to 32,767
float	4 bytes	3.4e-38 to 3.4e+38
double	8 bytes	1.7e-308 to 1.7e+308
void	-	-

Integer
- signed int (-32,768 to 32,767)
- unsigned int (0 to 65535) ↑ Range
  - → short int
  - → int
  - → long int

Real data-type :- Float, double

⇒ Derived Data-type :-

- The data-types which are derived from the Primary data-type are known as derived data-type.
- Ex :- Array, Pointers

⇒ User defined data-type :-

- The data-types which are defined by user are known as user defined data-types.
- Ex :- functions, structures

Operators :-

⇒ List out the various operators & explain with example :-

1. Airthmetic Operators
2. Relational operators
3. Logical Operators
4. Assignment Operators
5. Increment & Decrement Operators
6. Conditional Operators
7. Bitwise Operators
8. Special Operators
   - ↳ Size of Operators
   - ↳ Comma Operator

1. Airthmetic Operators :-

⇒ The airthmetic Operator is a binary operator, which requires two operands to perform its operation of airthmetic.

Operator	Description
+	Addition
−	Subtraction
*	Multiplication
%	Modulo
/	Division

Examples:-

WAP to Perform Addition, Subtraction, multiplication, Division & Modulo of two nums:-

```
Void main()
{
    int a,b, add, sub, mul, div, mod;
    pfc"Enter a & b:");
    sfc"%d %d", &a, &b);
    add = a+b;
    sub = a-b;
    mul = a*b;
    Div = a/b;
    mod = a%b;
    pfc"Addition =%d", add);
    pfc"\n Subtraction =%d", sub);
    pfc"\n Multiplication = %d", mul);
    pfc"\n Division = %d", div);
    pfc"\n Modulo = %d", mod);
    getch();
}
```

O/P:-

Enter a & b: 10 5
Addition = 15
Subtraction = 5
Multiplication = 50
Division = 2
Modulo = 0

⇒ CAP to enter days as an input & convert it into years, months & days.
(Consider 1 year = 365 days
1 month = 30 days)

↦
```
Void main()
{
    int days, y, m, d;
    pfc("Enter days:");
    sfc("%d", &days);

    y = days/365;                    // year
    m = (days % 365)/30;             // month
    d = (days % 365) % 30;           // Remaining days

    pfc("%d years %d months %d days",
                    yes y, m, d);
    getch();

}
```

↦ O/p :-

Enter days: 500
1 years 4 months 15 days

⇒ CAP to swap two numbers with using arithmetic operators:-

↦
```
Void main()
{
    int a, b;
```

Page No
Date

```
pf c"Enter a & b:");
sf c"%d %d", &a, &b);
```

a = 10, b = 5

```
a = a+b;
b = a-b;
a = a-b;
```

// a = 15
// b = 15 - 5 = 10
// a = 15 - 10 = 5

```
pf c"After swapping a = %d, b = %d",
                    a, b);
getch();
}
```

⇒ O/p :-

Enter a & b: 10 5

After swapping a = 5, b = 10.

(You can write any of the one example).

2. Relational Operators:-

↦ When we want to compare two values then we can use relational operators.

Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not Equal to

↳ Example:- 5

WAP to find out max number among two ~~operate~~ numbers.

```
↦ void main()
  {
      int a, b;
      pf("Enter two nums:");
      sf("%d %d", &a, &b);

      if (a>b)
      pf("a is max");
      else
      pf("b is max");
      getch();
  }
```

↦ O/P:-

Enter two nums:-
10 5
a is max.

3. Logical Operators:-

↦ The logical operators are used when we want to test more than one condition & make decisions.

Operator	meaning
&&	- Logical AND
||	- Logical OR
!	- Logical Not

↦ Example:-

↦
```
Void main()
{
    int a=15, b=2, c=25;
    if( a>b && a>c )
    Pf("a is max");
    else
    Pf("b or c max");
    getch();
}
```

↦ O/p :- b or c max.

(When we use && operator then if both conditions are true then true part will be executed, when we use || operator then if one condition is true then true part will be executed.)

Ex :- If we write :-

```
if (a>b || a>c)
     ↑      ↑
   15>2   15>25
    ↑      ↑
   true   false
```

but here, we will give o/p "a is max" because one condition is true.

4. Assignment Operators :-

↦ Assignment operators are used to assign the result of an expression to a variable.

↦ In addition, C has a set of 'shorthand' operators of the form

```
v op = exp;
```

v - variable
op - operator
exp - expression

↦ Example:-

Statement with ↓ shorthand Operator

a = a+1	→	a += 1
a = a-1	→	a -= 1
a = a*(n+1)	→	a *= n+1
a = a/(n+1)	→	a /= n+1
a = a%b	→	a %= b

" = " → Assignment Operator

5. Increment & Decrement Operators:-

++ and --
↑ ↑
Increment Decrement

↦ The operator ++ adds 1 to the operand; while -- subtracts 1.

↦ These operators have two types:-

++m	→	Pre increment
m++	→	Post increment
--m	→	Pre decrement
m--	→	Post decrement

↦ We can use these operators in the loop.

↦ Example :-

```
↦ void main()
  {
      int a= 10, b=5, x, y;
      x = ++a;
      y = b++;
    p.f c" x= %d, a= %d, y= %d, b= %d",
                          x, a, y, b);
      getch();
  }
```

↦ O/p :-

x = 11, a = 11, y = 5, b = 6

↦ A prefix operator first adds 1 to the operand & then the result is assigned to the variable on left.

↦ On the other hand, a postfix operator first assigns the value to the variable on left & then increements the operand.

6. Conditional Variable :- (Ternary Operator)

↦ A ternary operator pair '?:' is available in C to construct conditional expressions of the form :-

exp 1 ? exp 2 : exp 3;

↦ This exp is equivalent to :-

```
Condition ? TruePart : false Part ;
```

↦ Example :-

1. WAP to find out max number among two nums using conditional Statement.

↦
```
void main()
{
    int a=5, b=10, x;
    x= (a>b) ? a : b ;
       Condition  T  F
    Pf("Max num = %d", x);
    getch();
}
```

↦ O/P :-

max num = 10

2. WAP to find out max number among three nums.

```
Void main()
{
    int a=15, b=10, c=5, x;
```

```
x = (a>b) ? (a>c) ? a:c : (b>c) ? b:c;
pfc "max num = %d", x);
getch();
}
```

↳ O/P :-

max num = 15

7. Bit-wise Operator :-

↦ C has a distinction of supporting special operators known as "Bit-wise operators" for manipulation of data at bit level.

↦ These operators are used for testing the bits or shifting them right or left.

↦ Bitwise operators may not be applied to float or double.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift right

a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Ex:-

a = 2 → 0010
b = 3 → 0011

a & b → 0010 → (2)
a | b → 0011 → (3)
a ^ b → 0001 → (1)

a << 1 → 0010 → 0100 → (4)

a >> 1 → 0010 → 0001 → (1)

⇒ Example:-

WAP to use bit-wise Operators:-

⇒
```
Void main()
{
    int a = 2, b = 3;
pf("Bitwise AND = %d", a&b);
pf("\n Bitwise OR = %d", a|b);
pf("\n Bitwise XOR = %d", a^b);
pf("\n Bitwise shift left of a = %d", a<<1);
pf("\n Bitwise shift right of a = %d", a>>1);
  getch();
}
```

↳ O/P :-

Bitwise AND = 2
Bitwise OR = 3
Bitwise XOR = 1
Shift left of a = 4
Shift right of a = 1

8. Special Operators :-

(i) Sizeof Operator :-

The sizeof is a compile time operator & when used with an operand, it returns the number of bytes the operand occupies.

```
sizeof (variable name);
```

Example :-

↳

```
void main()
{
    int a;
    float b;
    char c;
    double d;

Pfc" Size of a = %d, b = %d, c = %d, d = %d",
  sizeof(a), sizeof(b), sizeof(c), sizeof(d));
 getch();
}
```

↳ O/P :-

Size of a = 2, b = 4, c = 1, d = 8

Page No.
Date

(ii) Comma Operator:-

↦ The comma operator can be used to link the related expressions together.

↦ A comma-linked list of expressions are evaluated left to right & the value of right most expression is the value of the combined expression.

Example:-

```
Value = (x = 10, y = 5, x + y);
```

Comma Operator

★ Type Conversion:-

↦ Type conversion is converting one data type into another.

↦ There are two types:-

1. Implicit (Automatic) Conversion
2. Explicit (forced type) Conversion

⇉ Implicit Conversion:-

↦ It is done by compiler on its own.

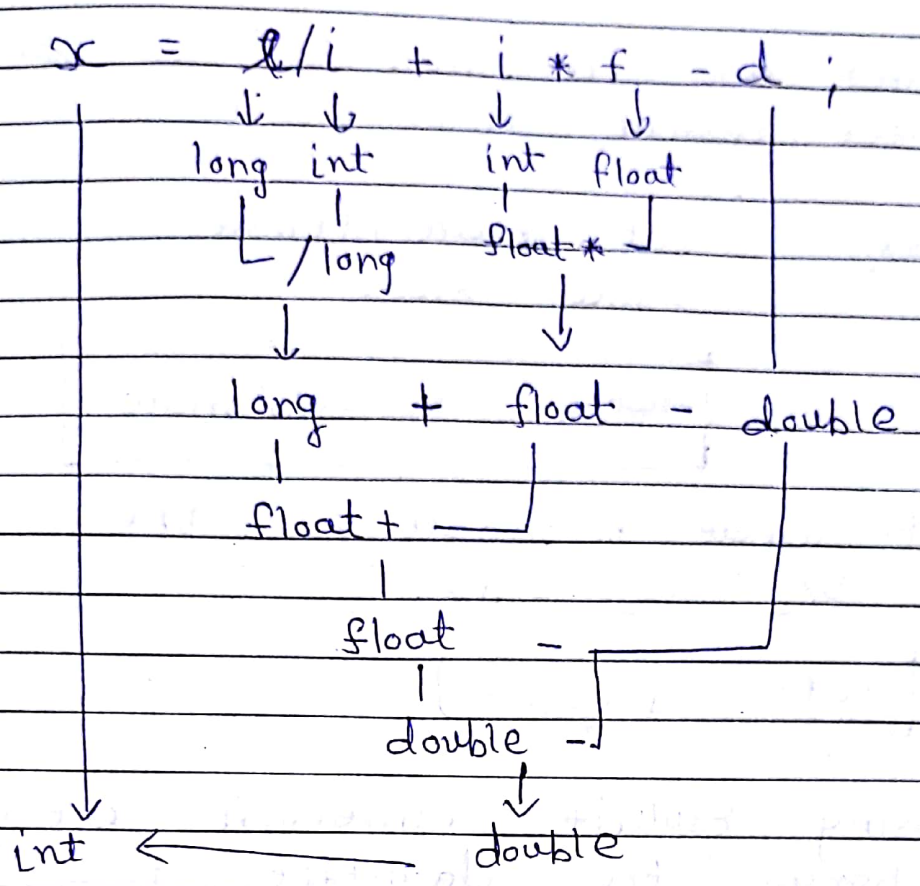
↦ Automatic Conversion is done from lower data-type to higher data-type.

↦ C automatically converts any intermediate values to the proper type, so that the expression can be evaluated without losing any significance.

Short int → int → long int → float → double

Ex:-

```
int i, x;
float f;
double d;
long int l;
```



↦ Explicit Type Conversion:-

↦ Force fully converting from one data-type to another by user. This process is known as forced type conversion or type-casting.

↦ The general form

```
(datatype) expression
```

↦ Ex:- if we want to find ratio of female & male numbers & the data-types of male & female are integer then int/int, we will

get ans in int. So, we couldn't get correct ans.

Ex:- int male, female;
float ratio;

```
ratio = female/male;
```

if female = 5, male = 3 then
5/3 = (1) → int

```
ratio = 1.000000
```

Using Explicit Conversion we can change the data-type for only the statement of exp. & will get correct ans.

```
ratio = (float) female/male;
```

Now, 5.0/3 = 1.66667

```
ratio = 1.66667
```
← Correct ans.

→ Precedence of Arithmetic Operators:-

High priority - * / %
Low priority - + -

→ The basic evaluation procedure includes 'two' left-to-right passes through the expression.

→ During the first pass, the high priority operators (if any) are applied as they are encountered.

→ During second pass, the low priority operators (if any) are applied as they are encountered.

Ex:- $X = a - b/3 + c*2 - 1$

When, $a = 9$, $b = 12$, $c = 3$

$X = 9 - 12/3 + 3*2 - 1$

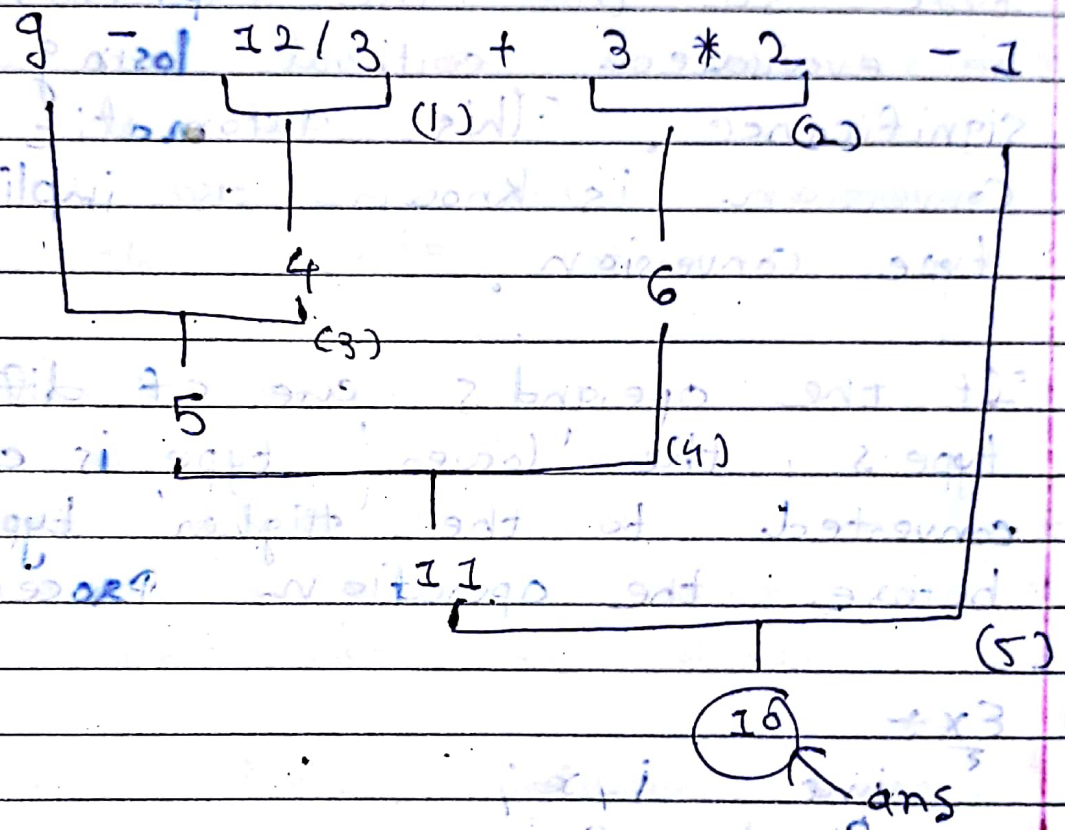
→ First Pass:-

1.) $X = 9 - 4 + 3*2 - 1$

2.) $X = 9 - 4 + 6 - 1$

Second Pass :-

Step 3: X = 5 + 6 - 1
Step 4: X = 11 - 1
Step 5: X = 10



⇒ Associativity :-

- Left to Right - ( ) , * , + , << , <
- Right to Left - = , Conditional operator
  >> , >