Module 1

Introduction to C Program → Part 1

* Introduction to C language
* Algorithm & flowchart
* Structure of C program
* C Tokens & data types

* Introduction to C language :-

C is a high level language. It was developed at the Bell Laboratories of USA in 1972. It is the outcome of efforts of Dennis Ritchie & Brian Kernighan.

Characteristic of C

* C language became popular because of the following reasons.

- C is a robust language, which consists of number of built-in functions & operators which can be used to write complex programs.
- C program is easy to learn.
- It helps in development of system software.
- No rigid format. Any number of statements can be typed in a single line.
- C is a structured programming language

- portability - C program can be run on different machines with little or no modifications.

Applications of C

C is used to develop the system as well as application software.

System softwares like,
- operating systems
- Interpreters
- Compilers
- Loaders
- Linkers
- Editor

Application softwares like,
- graphic package
- CAD applications
- Scientific & Engineering applications.

Character set

* The characters are used to form words, numbers & expressions depend on the computer on which the program is run.
* The characters in C are grouped into the following categories.

1. Letters (Alphabets)
2. Digits
3. Special characters
4. White spaces

Letters (Alphabets) → Upper case A ---- Z
Lower case a ---- z

Digits → All decimal digits 0 ---- 9

Special characters →

, comma	% percent sign
. period	& ampersand
; semicolon	- minus
: colon	+ plus
? question mark	[Left bracket
' apostrophe	] right bracket
! exclamation mark	{ left brace
/ Slash	} right brace
\ Black slash	
(left parenthesis	
) right parenthesis	

White spaces → Blank space
Horizontal tab
New line etc.,

* Algorithm & flowcharts :-

*Algorithm & flowcharts are the 2 import ant tools used in problem solving using digital computer.

* Algorithm & flowcharts & pseudo codes are some of the design techniques.

Algorithm

* Algorithm is a problem solving technique.
* An algorithm can be defined as a step-by-step procedure to solve a particular problem.
* It consists of english like statements. Each statement must be well defined to perform a specific operation. When these statements are executed for a given set of conditions, they will produce the required results.

Characteristics of algorithm

Each & every algorithm is characterised by the 5 important characteristics:

1. Input: It may accept zero or more inputs.
2. Output: It should produce at least one output (result).
3. Definiteness: Each instruction must be clear, well-defined & precise. There should not be ambiguity.
4. Finiteness: It should be a sequence of finite instructions.
5. Effectiveness: It means that the operations must be simple & carried out in a finite time.

Algorithm notations

While writing algorithm the following notations are considered.

1. Name of algorithm: It specifies the problem to be solved.
2. Step number: It is an identification tag of instruction. It is an positive integer.
3. Explanatory comment: It follows the step number & describes the operation. It should be written within a pair of square brackets.
4. Termination: It specifies the end of algorithm. It is generally "STOP" statement & last instruction of algorithm

Example 1: Write an algorithm to compute area of a circle.

Algorithm: Area of circle
Step 1: Start
Step 2: Read radius
Step 3: [compute the area]
Area = 3.142 * radius * radius
Step 4: [print the area]
print "Area of circle =", Area
Step 5: [End of algorithm]
Stop

Example 2: Write an algorithm to perform the basic airthmetic operations such as addition, subtraction, multiplication & division.

Algorithm: Airthmetic operations

Step 1: Start
Step 2: [Read value of A & B]
Step 3: [Compute sum, difference, product & quotient]
Sum = A+B
diff = A-B
prod = A*B
quot = A/B
Step 4: [print the sum, diff, prod & quot]
print 'Sum of A & B =', sum
print 'Difference of A & B =', diff
print 'product of A & B =', prod
print 'Quotient of A & B =', quot
Step 5: [End of algorithm]
Stop

Flowchart

* This is a chart showing a flow of logic involved in solving a problem. This is defined for an algorithm.

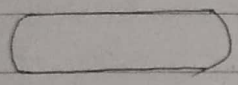
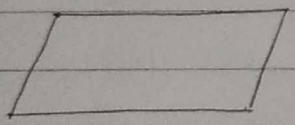
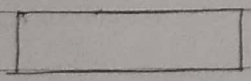
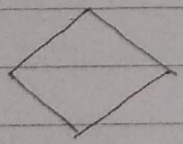
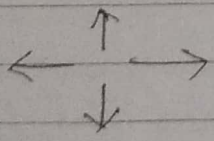
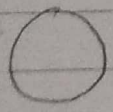
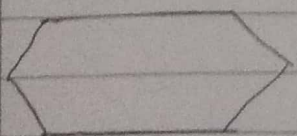
* The flowchart can be defined as a diagramatic representation of an algorithm.

* It is also referred to as the blueprint of an algorithm.

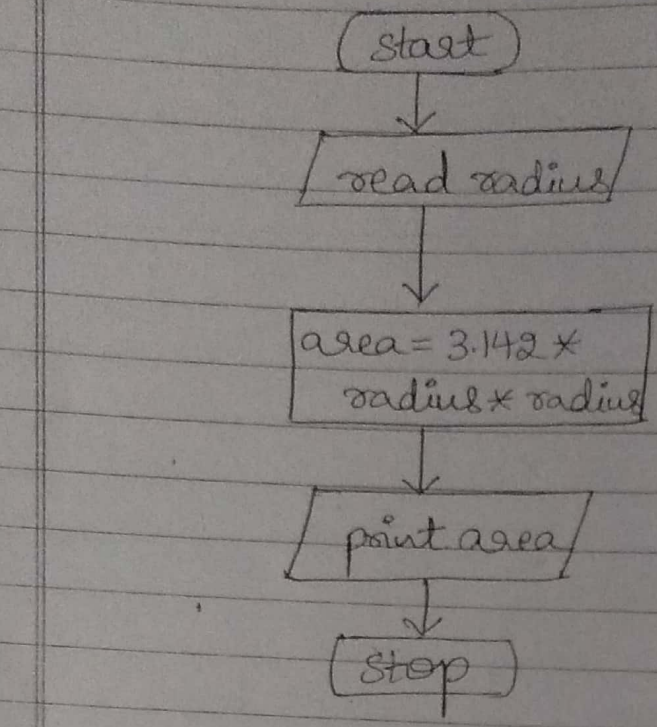
* The flowchart is an easy way to understand & analyze the problem. It is a useful aid for programmers & system analysts.

*Flowcharts are classified into 2 types.
1. Program flowcharts
2. System flowcharts.

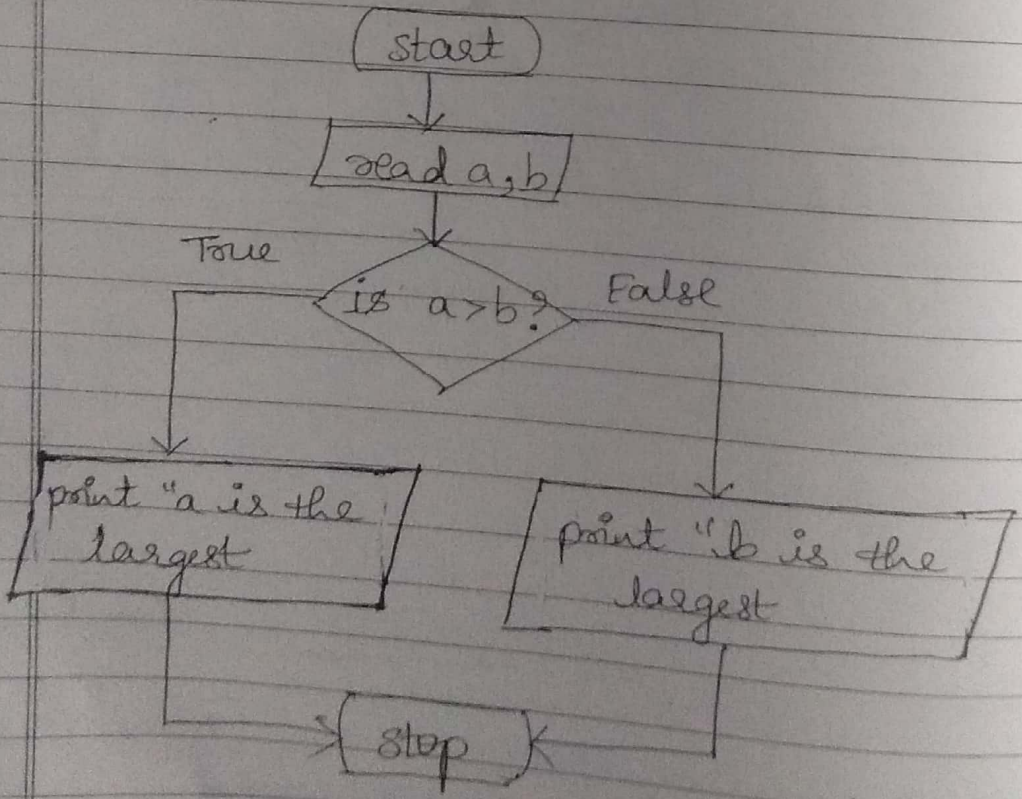
* Flowcharts make use of geometrical figures, to specify a particular operation. The below table shows different geometrical figures used in a program flowchart with their functions.

Geometrical figure	Name	Function
	oval	Start & Stop
	parallelogram	Input & output
	Rectangle	processing
	Diamond	Decision making
	Arrows	Connections
	Small circle	continuation
	Hexagon	Repetition / Looping

Example 1 : Draw a flowchart to find the area of circle when its radius is given.



Example 2 :- Draw a flowchart to find the largest of 2 numbers.



* Structure of C program :-

The complete structure of a C program is shown below

```
comment section
preprocessor statements / Inclusion section
global declarations;
main()
{
    declarations; → (Local variable)
    statements;
}
user defined functions
```

* preprocessor statements

These statements begin with # symbol & are also called the preprocessor directives. These statements direct the c preprocessor to include header files & also symbolic constants into a C program.

Some of the preprocessor statements are given below.

#include<stdio.h> : for std input/output functions

#define NULL 0 : for defining symbolic constant, NULL=0

* Global Declarations

Variables or functions whose existence is known in the main() function & other user defined functions are called as the

global variables (of functions) & their declarations are called global declarations.

* The main() functions

* the execution of a C program starts with main(). No C program is executed without the main() function. It calls other library functions & user defined functions.
* The left brace ({) indicates the beginning of the main() function. The right brace indicates the end of the main() function.

* Declarations

It is a part of C program where all the variables, arrays, functions etc. used in the C program are declared & may be initialized with their basic data types.

* statements

* these are instructions to the computer to perform some specific operations. They may be input-output ~~functions~~ statement, arithmetic statements, control statements. They also include comments. Comments are explanatory notes on some instructions.
* the statements to be commented must be enclosed within /* and */.

User-defined functions

* These are subprograms. It contains a set of statements to perform a specific task. These are written by the user, hence it is named as user-defined functions.

A sample C program is given below:

```
#include<stdio.h>
void main()
{
    printf("welcome to C\n");
}
```

Output
Welcome to C

* The 1st line tells the compiler to include the standard input/output header file to perform the reading & printing of data.

* The 2nd line is the main(),

* The body of C program contains only 1 statement i.e.,

printf("welcome to C\n");

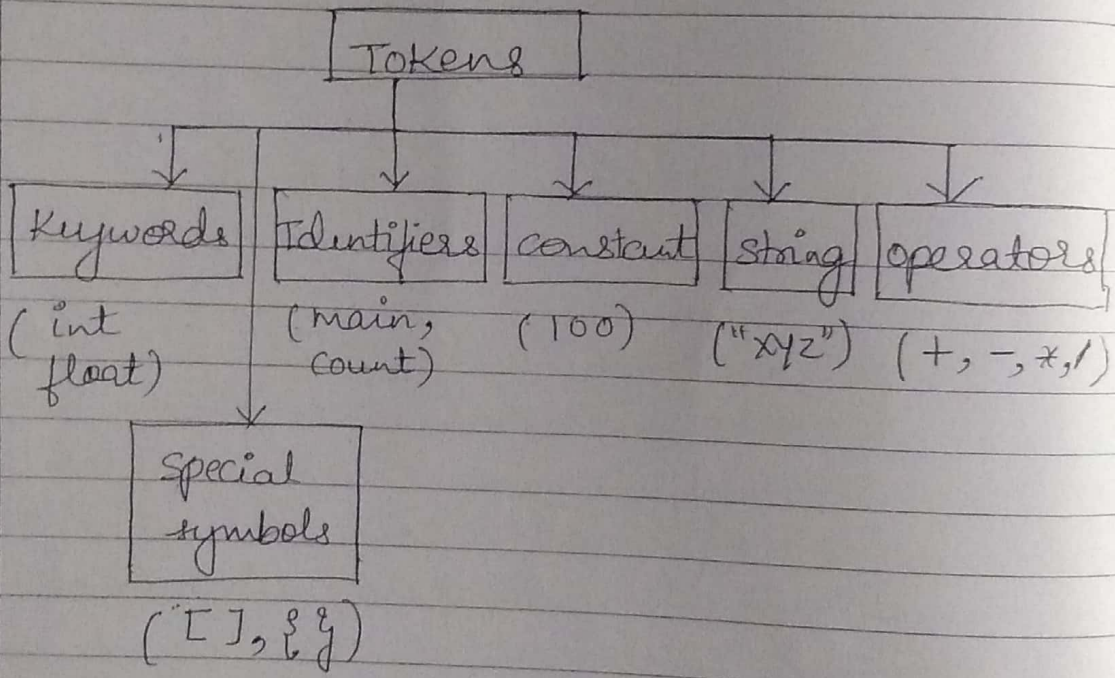
* When this statement is taken for execution, main() calls the printf()

function, which is included in <stdio.h>. This printf() prints the "Welcome to C" on the computer screen.

* C Tokens :-

It is the basic & the smallest units of a C program.

* There are 6 types of tokens in C.



* Keywords & Identifiers

* Each words in a C program is either a Keywords or an Identifier.
* All Keywords are the sequence of characters that have 1 or more fixed meanings.
* All Keywords should be written in lower letters.

P.T.O

Eg:-

auto	break	char	const	continue
double	else	if	float	int
void	while	switch	short	return

* Identifiers

* It is the name given to the program elements such as variables, array & functions
* It is the sequences of alphabets & digits.

Rules for forming Identifier names

* The first character must be an alphabet (uppercase or lowercase) or an underscore
* No special characters are allowed.
* No 2 successive underscores are allowed.
* Keywords should not be used as an identifiers.

* Datatypes :-

Datatypes indicate the type of data that the variable can hold. The data may be numeric or non-numeric in nature.

It is classified into -

1. Built-in (Basic) data type

2. Derived data type
3. user-defined data type

1. Built-in data type

i) Integer
ii) Floating point number
iii) Double precision
iv) Characters

In C, there is one & only one non-specific data type called void. It does not specify anything.

Types	Keyword	size (Bytes)
Integer	int	2
Floating-point	float	4
Double precision	double	8
characters	char	1

non-specific void — —

i) int :- This is a keyword used to indicate an integer number

* Any integer number is a sequence of digits without decimal point.

* The range of int depends on the word length of computer.

* Word length is the number of bits that can be accessed at a time by the processor.

* The 16-bit computer handles the integer from -32,768 to +32,767.

ii) float :-

* This is a keyword used to indicate a floating-point number.

* It is same as the real numbers.

* It is called as the floating-point numbe because the decimal point is shifted either to the left or to the right of some digits during manipulation.

* The range of float is -3.4E+38 to +3.4E+38.

* It has 6 digits of precision.

iii) double :-

* This keyword is used to indicate the double-precision floating-point number.

* It stores the 16 significant digits after the decimal point.

Eg :- 234.0000000000000000

* The size of double is 8 bytes

* the range is from

+1.7E-308 to +1.7E+308.

iv) char :-

* This keyword is used to indicate the characters

*. The size of char is 1 byte

* The range is from -128 to +127

* A data may be a character constant or string constant.

* character constant is a single character enclosed within a pair of apostrophe

Eg:- 'a', 'p' ---etc.,

* String constant is set of characters closed within a pair of double quots

Eg:- "computer"

*Data type modifiers

The basic data types except void can be modified using a series of type modifiers to fit the requirement of a particular program more closely.

These modifiers are also called qualifiers They are:

1. Signed
2. unsigned
3. Short
4. Long

*Signed :- * It is applied to integer variable
* It can be applied with a character data type to create small integer.

* Size of signed char is 1 byte

* The range is -128 to +127

* unsigned :-

*This can be used for both int and char

*. It is used to create a unsigned integer.

* Size of unsigned char is 1 byte

* Range is 0 to 255.

* long :-

*This is applied to int data type.

* Size of long int is 4 byte

Short:-

* This makes the size of an int half.

* Size of short int is 2 bytes

2. Derived data type:-

It includes
1. Pointer types
2. Array types
3. Structure types
4. Union types
5. Function types

* Variables

* The quantity that changes during the execution.

* The variables represent a particular memory location where the data can be stored.

* It is used to denote constants, functions, arrays etc.,

Eg:- Sum, area, age, length etc.,

* Rules for forming variable names

1. The 1st character of a variable name must be an alphabet or an underscore (_).

2. All succeeding characters consists of letters & digits.

3. Keywords should not be used as variables.

4. Special characters are not allowed.

* Declaration of variables

* The purpose of declaring variables is to reserve the amount of memory required for these variables.

Syntax is

Data_type	varlist	semicolon

where,

Data_type → is a basic data type such as int, float, char or double.

varlist → one or more variables of type data_type. & these variables must be separated by commas.

semicolon → a delimiter of this declaration

Eg:-
```
int length;
float area;
char ch;
int x, y, z;
```

* Assigning values to variables

* The process of giving values to variables is called the assignment of values.

* The assignment operator '=' is used to assign a value to a variable.

* Syntax is

| variable_name | = | value | ; |

where,

variable_name ⇒ the memory location where the 'value' should be stored.

= ⇒ assignment operator.

value ⇒ is a constant or a variable.

there are 2 methods :-

1. Initial values can be assigned to variables within the declaration section.

Eg :- int x = 1 ;
char ch = 'y' ;

2. Initial values are assigned to the variables in the executable part of a program.

Eg :- x = 10 ;
ch = 'y' ;

programs on data types

1) Write a C program to print a message.

```
#include<stdio.h>
void main()
{
clrscr();
printf("Hello world");
getch();
}
```

2) Write a C program to display the message line by line.

```
#include<stdio.h>
void main()
{
clrscr();
printf("Hello \n");
printf("How are you");
getch();
}
```

3) write a C program to print the integer value.

```
#include<[redacted]stdio.h>
void main()
{
int a;
clrscr();
a=10;
```

```
printf(" The a value is %d", a);
getch();
}
```

4) Write a C program to perform the addition of 2 integers.

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

5) Write a C program to print the integers value.

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

6) Write a C program which accepts the integers & prints the integer's value.

```
#include<stdio.h>
void main()
{
  int a,b;
  clrscr();
  printf("Enter a value\n");
  scanf("%d",&a);
  printf("Enter b value\n");
  scanf("%d",&b);
  printf("The value of a is %d and b
  value is %d",a,b);
  getch();
}
```

7) Write a C program which accepts the integer & peeforms the addition of integers & print the sum.

```
#include<stdio.h>
void main()
{
  int a, b, sum=0;
  clrscr();
  printf("enter values of a and b\n");
  scanf("%d%d",&a,&b);
  sum=a+b;
  printf("The sum is %d",sum);
  getch();
}
```

8) Write a C program to find the area of circle

Formula

$area = \pi r^2$

area = pi * r * r or pi * pow(r,2)

program :-

```
#include<stdio.h>
#include<math.h>
#define PI 3.142

void main()
{
  float r, area;
  clrscr();
  printf("enter value of radius\n");
  scanf("%f", &r);
  area = pi*r*r;
  printf("area of circle is %f", area);
  getch();
}
```

9) Write a C program to find the area of rectangle.

Formula

area = b x h

program

```
#include<stdio.h>
#include<math.h>
void main()
{
    int b, h;
    float int area;
    clrscr();
    printf(" enter the value of b and h");
    scanf("%d%d", &b, &h);
    area = b*h;
    printf(" area of rectangle is %d", area)
    getch();
}
```

10) Write a C program to find the area of triangle.

```
#include<stdio.h>
#include<math.h>
void main()
{
float a, b, c;
    float s, area;
    clrscr();
    printf("enter the value of a, b and c\n");
    scanf("%d%d%d", &a, &b, &c);
    s=(a+b+c)/2.0;
    area=sqrt(s*(s-a)*(s-b)*(s-c));
    printf("area=%f", area);
    getch();
}
```

11) Write a C program to accept characters, integer & floating point number & display the same.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 char ch;
 int a;
 float b;
 clrscr();
 printf("enter a character\n");
 scanf("%c",&ch);
 printf("enter a number\n");
 scanf("%d",&a);
 printf("enter a floating point number\n");
 scanf("%f",&b);
 printf("character is %c",ch);
 printf("integer is %d",a);
 printf("Floating point number %f",b);
 getch();
}
```

12) Write a C program to accept your name & display the same.

```
#include<stdio.h>
void main()
{
 char ch[100];
 clrscr();
 printf("enter your name\n");
 scanf("%s",&ch);
```

a → %c

```
printf ("name is %s", ch);
getch();
}
```

13) write a C program to swap 2 number using temporary variable.

```
#include<stdio.h>
void main()
{
  int a, b, temp;
  clrscr();
  printf (" enter the value of a & b \n");
  scanf("%d%d", &a, &b);
  temp = a;
  a=b;
  b= temp;
  printf(" After swapping a is %d \n b is %d
                                    a,b);
  getch();
}
```

14) write a C program to swap 2 number without using temporary variable.

```
#include<stdio.h>
void main()
{
  int a, b;
  clrscr();
  printf (" enter the value of a & b \n");
  scanf("%d%d", &a, &b);
  a = a+b;
  b=a-b;
```

```
    a = a-b;
    printf("after swapping a is %d \n
        b is %d", a, b);
    getch();
}
```

Tracing a=10 b=5

a=a+b = 10+5 = 15

b=a-b = 15-5 = [10] → b

a=a-b = 15-10 = [5] → a

Finally a=5 b=10

Possible question

1) what is a algorithm? Explain with example.
2) Define flowchart? Explain with example.
3) Explain the structure of C program
4) Programs on Any datatypes
5) Explain data types?

* Program :- It is a set of instruction/ statements given to the computer to achieve specific task.

* Statement :- Smallest element of a programming language.

6) what is variable? Explain declaration & initialization of variables?

Module 1 (part 2)

operators and Expression → part 2

* 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 valu...

*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 operators

- 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 operator.

```
#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 logi operations on the given expression.
An expression containing logical operator returns 0 or 1 depending upon whethe 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 operators

* 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();
  print 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 betn 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
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:- ~12 = -13

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

For negative numbers

n = -12 → ~-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]
```

Eg: 12 & 25

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

12 & 25 = 8

b) Bitwise OR operator

* The output of bitwise OR is 1 if one corresponding bit of 2 operand

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 oper 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 (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 and 0 to 1.

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

$$\sim 0 = 1$$

$$\sim 1 = 0$$

Eg:- ~12 = -13

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

For negative numbers

n = -12 → ~-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
                 11 -> 2's
1111 1111 1111 0100 -> -12
```

$\sim -12 =$ 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-bit

60 = 00111100

13 = 00001101

Output

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

* Operators precedence & Associativity

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 expressions 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();
}
```

Lab program

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.
```

(or)

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

```
clrscr();
printf("enter value of a, b and c\n");
scanf("%d%d%d", &a, &b, &c);
d=(a>b)? a:b;
e=(d>c)? d:c;
printf("Largest number is %d", e);
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	L to R
?:	10	R to L
=, +=, -=, *=, /=, %.=	11	R to L
,	12	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

13 + 3 - 6
└─┘
16 - 6
└──┘
= 10

3) 5 + ((6 - 3) * (5 - 1))
└─┘
5 + (3 * (5 - 1))
└─┘
5 + (3 * 4)
└──┘
5 + 12
└──┘
= 17

Program :-

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a;
  clrscr();
  a=100/20<=10-5+100%10-20==5>=1
  printf("a=%d",a);
  getch();
}
```

* Expression :- Sequence of operands & operators that reduces to a single value

* 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 promotio
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

More example for implicit type conversion

2)
```
#include<stdio.h>
void main()
{
    int x = 10;
    char y = 'a'; float z;
    x = x + y; // 'a' = 97 (ASCII value)
                  y is implicitly converted to int
    float z = x + 1.0; // x is implicitly converted to float.
    printf("x = %d \t z = %f", x, z);
    getch();
}
```

output :- x = 107, z = 108.000000

3)
```
#include<stdio.h>
void main()
{
  clrscr();
  printf("%d",'A');
  printf("%d",'a');
  getch();
}
```

Output :- 65 97

* Convert mathematical expression into C-Equivalent Expressions

1) $Area = \pi r^2 + 2\pi r h$

$Area = \pi * r * r + 2 * \pi * r * h$

2) $Area = \sqrt{s(s-a)(s-b)(s-c)}$

$Area = sqrt(s*(s-a)*(s-b)*(s-c))$

possible questions

1) what is an operator? Explain the different types of operator based on the operation
2) Explain the different types of operator based on the number of operands?
3) Explain type conversion with an example?
4) Evaluate the following expression and find the value of a, b, c, p, q & r?

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