PAGE NO.:
DATE: / /

• Note:
There is no exponent operator in C. There is a readymade library function Power (pow) to carry out exponential.

for eg: 5^4 = pow(5,4).
a=7 b=3
a+b = 10
a-b = 4
a/b = 2
a%b = 1
a*b = 21

★ Relational Operator:
There are 6 relational operators of they are

operator	purpose
<	less than
>	greater than
<=	less than equal to
>=	greater than equal to
==	equal to
!=	not equal to

The relational operator relates the 2 equation of gives the value in the form of truthness or falseness i.e., true or false (or) 1 or 0.
for eg: Suppose i, j, K are 3 integer variables then the expression

PAGE NO.:
DATE: / /

i=1 , j=2 , K=3

Expression	values	
i<=j	True	1
(i+j)>=k	True	1
(j+K)>(i+5)	False	0.
K!=3	False	0
j==2	True	1

★ Logical Operator:
There are 3 logical operators in C of they are

operator	purpose
&&	AND
||	OR
!	NOR

• Logical operators are operate on relational expressions which gives true or false values i.e., 1 or 0 or a bool value, they also called boolean expressions in other words we say that logical operators act upon operands that are themselves logical expression.
The result of logical AND operation will be true if both operands are true.
The result of logical OR operation will be true if either operand is true of both operands are true.

for eg: if i=7, f=5.5, c='w',
i) (i>=6) && (c=='w') ii) (i>=6) || (c=='g')
T && T T || T
T T.

PAGE NO.:
DATE: / /

03-01-19

★ The Conditional Operator :

Simple conditional operations can be carried out with the conditional operator (? :). An expression that makes use of the conditional operator is called a conditional expression.

Such an expression can be return in place of more tradition if-else statement. It can be return in form:
expression 1 ? expression 2 : expression 3.

When evaluating a conditional expression, expression 1 is evaluated first. If expression 1 is true then expression 2 is evaluated otherwise expression 3 is evaluated i.e., (When expression 1 is false)

Condition ? true part : false part.

Eg : i = -5
(i < 0) ? 0 : 100.

The expression i<0 is evaluated first, if it is true, the expression takes the value zero otherwise it takes the value 100.

Eg : age > 18 ? printf ("eligible") : print "(not eligible")

★ Unirary Operator : -, ++, --, * (pointer), etc.

Increment & decrement operator:

These are unirary operators in C operate on single operand. The increment operator increases the value of variable by 1 & the decrement operator decreases the value by 1.

PAGE NO.:
DATE: / /

for eg : i = 5;
++i; ⟶ This increment the value of i by 1 so now i becomes 6 ★

★ The following program illustrate the use of increment & decrement operator.

```
# include <stdio.h>
# include <conio.h>
void main ()
{
  int i,j;
  i = 5;
  j = 10;
  printf ("original value of i %d", i);
  ++i;
  printf ("\n value of i after increment: %d", i);
  printf ("\n value of j initially %d", j);
  --j;
  printf ("\n value of j after decrement %d", j);
  getch ();
}
```

original value of i 5
value of i after increment 6
value of j initially 10
value of j after decrement 9

PAGE NO.:
DATE: / /

```
/*
# include <stdio.h>
# include <conio.h>
  void main ()
{
   int i,j;
    i= 5;
    j= 10;
printf (" original value of i %d", i);
printf ("\n value of i after increment %d," i++);
printf ("\n value of j initially %d", j);
printf ("\n value of j after decrement %d", j--);
printf ("\n value of i and j %d %d", i, j);
  getch ()
}
```

★ Assignment operator :

This operator assign a value of expression to an identifier. Its general format is

identifier = expression;
↳ assingment op.

for eg: a = 3;
x = y;
area = length * breath;

• Note: Remember that the assignment operator (=) & the equality operator (==) are distintly different.

Multiple assignment is in the following form:

identifier 1 = identifier 2 = identifier 3 = expression;

for eg: x = y = z = 5 + 3;

PAGE NO.:
DATE: / /

This assignment are carried out from right to left. i.e., 5+3 get evaluated first that value is assigned to z, then the value of z, then the value of z assign to y & then to x.

C contains 5 additional assignment operator such as

+=, -=, *=, /=, %=

They are written in the form;

expression 1 += expression 2;

& it is equivalent to

expression 1 = expression 1 + expression 2;

eg: x += y ⟹ x = x+y

04-01-19

★ Input Output Statements :

C language is acomparied by collection of library func° with includes no. of input, output functions.

getch(), putchar(), get(), puts(), scanf() & printf() are the 6 function which permits the transfer of info between the computer & input, output devices (keyboard & monitor)

The getch() & putchar() function allow single character to be transferred into & out of the computer. Gets() & puts() function facilate the input output of strings. The scanf() & printf() functions permits the transfer of single characters, strings & numeric value.

Entering Input Data using the scanf() function :

Input data can be entered into the computer from a standard input device by means of the C library function scanf(). This function can be used to enter any combi-

nation of numerical values, single characters & strings. The function returns the no. of data items that have been entered successfully. The general form of scanf() statement is scanf ("control string", arg1, arg2..., argn);

arg1, arg2... & so on are argument that represent the individual input data item. (Actually the argument represent pointer, that ~~indicate~~ indicate addresses of data items within computer memory).

for eg:

```
scanf ("%d %d %d", &a, &b, &c);
```

Control string — address operator — arguments.

Control strings refers to string containing certain required formatting information. There is one character for each input data item & each character group must begin with '%' sign. following are the most frequently used control strings They are:

- %c → character
- %s → string
- %d → integer
- %f → float value
- %ld → long integer
- %u → unsigned integer
- %o → for octal value
- %x → for hexadecimal value.

★ Writing output data — The printf().

Output data can be written from the computer's standard output device using the library function printf(). This function can be used to output any combination of numerical values, single character & string. It is similar to scanf() function except that its purpose is to display data rather than enter into the computer. In other words we say that printf statement moves data from computer to output device. Its general format is:

printf("control string", arg1, arg2, ... argn);

eg: printf("%d %d %d", a, b, c);

08-01-19

★ The Control Statement:

We had seen C programmes in which instructions were executed in the same order in which they appear within the program.

Each instruction was executed once & only once. they do not include any logical control structures. Such programs are unrealistic.

A realistic C program may req. that logical test be carried out at some particular point within the program. One of several possible actions will then be carried out depending on the outcome of logical test. This is known as branching. There is also a special kind of branching called selection in which one group of statement is selected from several available group.

PAGE NO.:
DATE: / /

Sometimes program may require that group of instruction may executed repeatedly, until some logical condition has been satisfied. This is known as looping statement.

Sometimes the required no. of repetation is known in advance & sometimes the competition cont. indefinately until the logical condition becomes true.

• All these operation can be carried out using various control statement in C.

Branching - The if-else statement:

In some problem requires comparison between two no.s or two strings in such situation if-else statement carry out the solution for problem.

If instruction carries out a logical test & then takes one of the two possible action depending on the outcome of the test (outcome may be true or false).

1) if (expr)
   statement 1;

eg: if (x > 10)
   printf ("How good you are");

2) if-else → General format is

⇒ if (expr)
   do this 1;
⇒ else (expr)
   do this 2;

eg: if (marks > 35)
   printf ("Pass");
   else
   printf ("Fail");

PAGE NO.:
DATE: / /

Nesting:

1. Nested if → General format

```
if (expr1)
{
 if (expr 2)
  do this 1;
 else
  do this 2;
}else
{
 if (expr3)
  do this 3;
 else
  do this 4;
}
```

Eg:

```
if (nation == "India")
{
 if (age >= 18)
  printf ("Eligible for voting");
 else
  printf ("Not eligible for voting");
}
else
{
 if (nation == "Pak")
  printf ("Not allowed");
 else
  printf ("Allowed as tourist");
}
```

The general format of if statement shows that expression (expr) must be placed in parenthesis. The statement will be executed after if, if the expression has true value or non-zero value.

The following program illustrate this:

```
{
 int first, second;
 printf ("Enter two numbers");
 scanf ("%d %d", &first, &second);
 if (first > second)
  printf ("First is bigger");
```

```
if (second > first)
printf ("second is bigger");
if (first == second)
printf ("Both numbers equal");
getch();
}
```

09-01-18

Eg:
```
# include <stdio.h>
# include <conio.h>
void main() {
{
int percent;
printf ("Enter percentage");
scanf ("%f", &percent);
if (percent >= 75)
printf ("Distinction");
if (percent >= 60)
printf ("first class");
if (percent >= 50)
printf ("second class");
if (percent <= 40)
printf ("fail");
getch();
}
```

Eg:
```
# include <stdio.h>
# include <conio.h>
void main()
{
int number;
printf ("Enter number");
scanf ("%d", &number);
if (number > 0)
printf ("positive");
if (number < 0)
printf ("negative");
if (number = 0)
printf ("zero");
getch();
}
```

If-else :

The format of if-else statement is as follow.

```
if (expr1)
    statement 1;
else
    statement 2;
```

if expression 1 is true, then statement 1 will be executed otherwise statement 2 will be executed.

Eg:
```
void main()
{
int first, second;
clrscr();
printf ("Enter two number");
scanf ("%d %d", &first, &second);
if (first == second)
printf ("Both nos are equal");
else
printf ("Both no. are not equal");
getch();
}
```

Eg: Write a program to check the given no. is odd or even

```
# include <stdio.h>
# include <conio.h>
void main ()
{
  int num;
  clrscr ();
  printf (" Enter the number");
  scanf (" %d," &num);
  if (num % 2 == 0)
    printf (" no. is even");
  else
    printf (" no. is odd");
  getch ();
}
```

Eg: Write a program to check a given no. is divisible by 5

```
# include <stdio.h>
# include <conio.h>
void main ()
{
  int num;
  clrscr ();
  printf (" Enter the number");
  scanf (" %d," &num);
  if (num % 5 == 0)
    printf (" no. is divisible by 5");
  else
    printf (" no. is not divisible by 5");
  getch ()
}
```

Eg:

```
# include <stdio.h>
# include <conio.h>
void main ()
{
  int temp;
  clrscr ();
  printf (" Enter the temperature");
  scanf (" %d," &temp);
  if (temp >= 20)
    printf (" Temperature is too hot");
  pf (temp <= 10)
    printf (" Temperature is too cold");
  if (temp <20 && >10) (temp <20) && (temp >10))
    printf (" Temperature is moderate");
  getch ();
}
```

11-01-19

Q. Write a program to find smallest among three nos

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

```
printf (" Enter three no.s");
scanf (" %of %of %of ", &a, &b, &c);
 if (a<b)
  {
   if (a<=c)
    {
     printf (" \n smallest no. is :, "a);
    }
   else
    {
     printf (" \n smallest no. is :, "c);
    }
  }
 else
  {
   if (b<=c)
    printf (" smallest no. %f, " b);
   else
    printf (" smallest no. %f, "c);
  }
 getch ();
}
```

Q Write program to find biggest among three no.s.

```
# include <stdio.h>
# include <conio.h>
 void main ()
{
 int a,b,c;
 clrscr ();
 printf (" Enter three no.s");
 scanf (" %d %d %d ", &a, &b, &c);
  if (a >= b)
  {
   if (a >= c)
    {
     printf (" \n biggest no. is :, "a);
    }
   else
    {
     printf (" \n biggest no. is :, "c);
    }
  }
 else
  {
   if (b >= c)
    printf (" biggest no. %d, " b);
   else
    printf (" \n biggest no. %d, " c);
  }
 getch ();
}
```

Q Write a program that display GM, GA, GE if time is out of range if the time >= 0 && <=12

Good Morning

if time (>12 && <=18) → GA.

if time (>18 && <= 24) → GE

else

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int time;
  clrscr();
  printf("Enter the time");
  scanf("%d", &time);
  if((time >= 0) && (time <= 12))
    printf("Good Morning");
  if((time > 12) && (time <= 18))
    printf("Good Afternoon");
  if((time > 18) && (time <= 24))
    printf("Good Evening");
  else
    printf("time is out of range");
  getch();
}
```

12-01-19

Looping:

Suppose we want to display "Hello" on output screen 5 ~~ty~~ times in 5 different lines, we might think of writing printf statement 5 times consisting of a string "Hello\n" 5 times.

If we want to like to print hello 500 times the above procedure is quiet lengthy. In C programming there is a facility available to repeat certain works in the form of looping statement.

There are 3 types of loop statement:

1) while - loop
2) do-while loop
3) for loop.

★ The while statement:

It is used to carry out looping operations, in which group of statements executed reapeatedly until some condition has been satisfied.

The general form of statement is

```
while (expression)
{
  statement 1;
  statement 2;
}
```

The statement will be executed repeatedly as long as the expression is true. (non-zero value).

This statement can be simple or compound.

Eg:

```
count = 1;  → /* Initialization of loop counter */
while (count <= 5)  ← condition
{
  printf("Hello\n");
  count = count + 1;  /* increment statement */
}
```

Output:

Hello
Hello
Hello
Hello
Hello
Hello

1 to 100
print even nos from 100 to 200.

PAGE NO:
DATE: / /

Eg: /* Write a program to print 1 to 10 nos.
#include <stdio.h>
#include <conio.h>
void main ()
{
int i;
i=1;
while (i<=10)
{
printf ("\n%d,"i);
i++;
}
printf ("\n program over")
getch ();
}

Eg: /* Write a program to print 1 to 100 nos.
#include <stdio.h>
#include <conio.h>
void main ()
{
int i;
i=1;
while (i<=100)
{
printf ("\n%d,"i);
i++;
}
printf ("\n program over")
getch();
}

PAGE NO:
DATE: / /

Eg: /* Write a program to print even nos from 100 to 200.
#include <stdio.h>
#include <conio.h>
void main ()
{
int i;
i=100;
while (i<=200)
{
printf ("\n%d," i);
i+2;
}
printf ("\n program over")
getch ();
}

17-01-19

* The do-while statement:-

When a loop is constructed using the while statement the taste for continuation of the loop is carried out at the beginning of each pass. Sometimes it is desirable to have a loop with the taste for continuation at the end of each pass.

This can be accomplished by means of the do-while statement.

The general form of do-while statement:

do
{