Page No.
Date

Recursion

```
sum(int n)
{
   if (n==1)
     return 1;
   return n*+sum(n-1);
}
main()
{
  printf("\n Add.n of first 5 terms : %d", sum(5));
}
```

```
/*
Add.n of first 6 terms = 6 + Addition of first 5 terms
Add.n of first 5 terms = 5 + Addition of first 4 terms
                  .
                  .
                  .
                  1
*/
```

In Recursion, result of next term depends upon previous terms.

que. Calculate $6!+5!+4!+3!+2!+1! = ?$

```
mul(int n)
{
  if (n==1)
    return 1;
    return n*mul(n-1);
}
add (int n)
{
   pf ("+%d!", n);
   if (n==1)
      return 1;
   else
      return mul(n)+add(n-1);
}
main()
{
  pf ("=%d", add (6));
}
```

Charac
Alpha
a-z

Digits

Speci

Character Set

Alphabets

a-z, A-Z, _

Digits

0-9

Special symbols

i) operators

a) Unary

~ ! - + -- ++ & * sizeof

b) Binary

() [] . -> * / % + - << >>

< <= > >= == != & ^ | &&

|| = += -= *= /= %= <<= >>=

&= ^= |=

c) Ternary

?: ,

ii) Non-operators

" ' ' # : ; { } \ /* */

Tokens

Keywords | ~~Qualifiers~~ Identifiers | constants | operators

Keywords:

* Qualifiers:
  signed unsigned short long

* Data types:
  char, float int double void

* control statements
  * conditional:
    if else switch case default for while do

  * Unconditional:
    break continue goto return

* storage classes
  auto static extern register

* Types of pointers
  near far huge

* User-Defined datatypes
  struct union enum typedef

* other languages:
  pascal asm ada

* Miscelleneous:
  volatile sizeof

C is a case sensitive language except for constants

compile time error - checking syntax (grammer)

* Keyword + identifier + operator + constants

Page No. numeric

Identifiers

- Rules for defining an identifier

i) It should not be a keyword (grammer)

ii) It should not start with a digit (Reason - hexadecimal constant)

iii) Special symbols are not allowed.

iv) Spaces are not allowed.

v) Its minimum length should be one character.

ex. main()
{
int _=66;
printf("%d",_);
}
o/p = 66.

vi) Maximum length is compiler dependent.

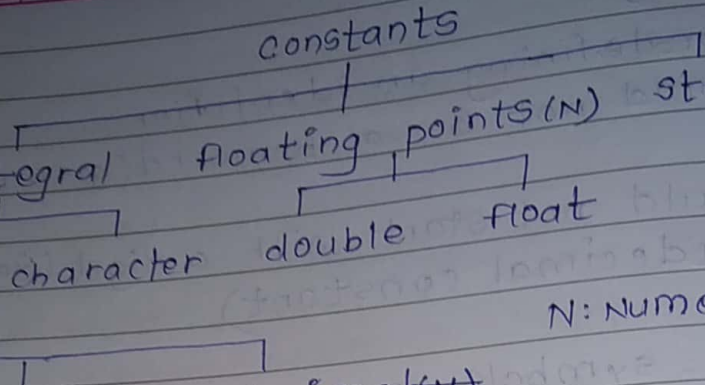
vii) Standard length considered as 8 characters. (In Java it is 256 character)

viii) In a same/single scope we cannot have two identifiers with the same name.

Scope ⇒ { to }

ex. main()
{
int x;
char x; // error: multiple declaration for 'x'.
}

In C there are 3600 [illegible]
In Java there are 75000 classes.



N: Numeric constants

Data Types:

Type	length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4*(10**-38) to 3.4*(10**+38)
double	64 bits	1.7*(10**-308) to 1.7*(10**+308)
long double	80 bits	3.4*(10**-4932) to 1.1*(10**+4932)

Character set

Alphabets

a-z , A-Z , _

Digits

0-9

Special Symbols

• Operators

- Unary ⇒ ~ ! + - ++ -- & * sizeof
- Binary ⇒ () [] . -> * / % + - << >> < <= > >= == != & ^ | && = += -= /= %= <<= >>= &= ^= |= ,
- Ternary ⇒ ?:

• Non-Operators

{} " " ' ' ; : \ #

C is a case sensitive language.
Except 1) Hexadecimal constant

Tokens

Keywords Identifiers constant operators

Keywords:
* Qualifiers:
signed unsigned short long
* Datatypes:
char int float double void
* control statements
- Conditional:
default if else switch case while do for

- Unconditional
break continue goto return

* storage classes
auto static extern ~~registe~~return

* Types of Pointers
near far huge

* other Languages:
Pascal asm ada

* Miscelleneous
sizeof volatile.

3 phases of Program

Design
compi
Run t

Rules

1. It
2. It s (R
3. Spe
4. Spa
5. Mi
6.

3 phases of program
- Design time = typing error
- compile time ⇒ syntactical errors (grammer)
- Run time ⇒ logical errors.

Page No.
Date

64
2 | 1
2 | 0 | 5

256
128
127
1111 1101

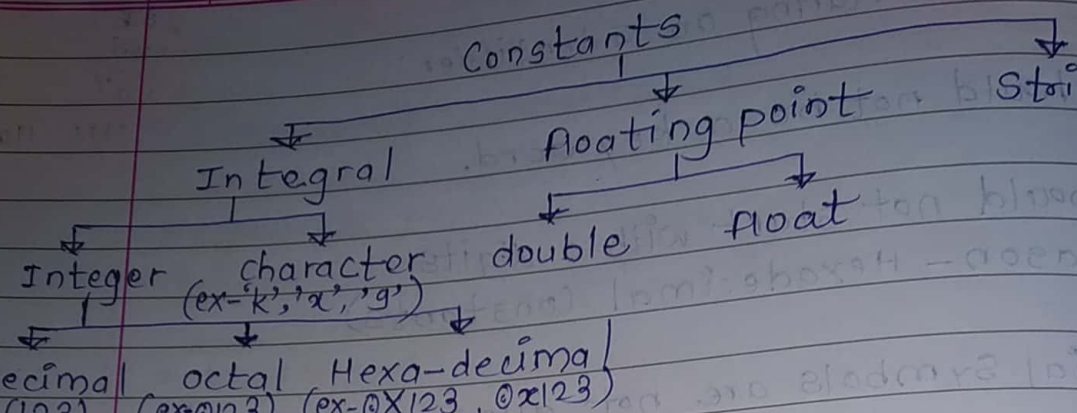
Rules for Declaring an Identifier:

1. It should not be a keyword.
2. It should not start with a digits.
   (Reason – Hexadecimal constants)
3. Special Symbols are not allowed.
4. Spaces are not allowed.
5. Minimum length should be one character.

```
main()
{
  int _=44;
  printf("%d",_);
}
```

o/p = 44

6. Maximum length is compiler dependent
   Std. length is considered as 8 characters.



Data Types.

Type	size	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,7
float	32 bits	3.4*(10**-38) to 3.4*(10**+38)
double	64 bits	1.7*(10**-308) to 1.7*(10**+30)
long double	80 bits	3.4*(10**-4932) to 1.1*(10**+49)

default qualifier = signed
default datatype = int

Page No.
Date

Syntax for Declaring a variable

[storage class] [qualifier] [type]
var1 [[= val1], var2 [[= val2], var3 [[= val3 ...]]]];

* main()
{
char x = 255;
printf("%d", x);
}
o/p = -1

1111 1111
0000 0001
-1

signed — sign bit — MSB
→ 0 = +ve NO. = Human understandable
→ 1 = -ve NO = Machine understandable

Unsigned data bit

main()
{
char x = 112;
printf("%d", x);
}

0111 0000 ⇒ 112
o/p = 112

main()
{
char x = 128;
pf("%d", x);
}
o/p = -128

```
main()
{
  unsigned char x=128;
  printf("%d", x);
}
```

o/p = 128

```
main()
{
  unsigned char x=-1;
  printf("%d", x);
}
```

o/p = 255

```
main()
{
  char x=-255;
  printf("%d", x);
}
```

o/p = 1

```
main()
{
  unsigned char x=512;
  printf("%d", x);
}
```

o/p ⇒ 0

```
main()
{
  unsigned char x=129;
  printf("%d",x);
}
```

o/p = 129

128	64	32	16	8	4	2	1	
1	0	0	0	0	0	0	1	
0	1	1	1	1	1	1	1	⇒ −127

ASCII chart

*	42	A=65	a=97
0	48	B=66	b=98
1	49	C=67	⋮
2	50	⋮	
⋮			z=122
⋮		Z=90	
9	57		

```
main()
{
  char a='9';
  printf("%d\n",a);
  printf("%c\n",a);
}
```

o/p ⇒ 57
9

* program for ASCII values.

```
main()
{
 int i;
 for(i=0; i<=255; i++)
  pf("%c : %d", i, i);
}
```

*
```
main()
{
  char x=3, y='3';
  printf("%d\n", x);
  printf("%dc\n", x);

  printf("%d\n", y);
  printf("%dc\n", y);
}
```

o/p : 3
♥
51
3

*
```
main()
{
   int x=128;
   int y=0128;
   int z=0x128;

  printf("x=%d\n", x); //128
  printf("x=%#o\n", x); //0178
  printf("x=%#x\n" x); //0x80

  printf("y=%d\n", y); //1112
  printf("y=%#o\n", y); // 0126
  printf("y=%#x\n", y); // 0x56
```

flag used for showing base at ~~ad~~ o/p.
decimal no. do not have base.
octal has base 0 (zero)
~~Hexadecimal base is 0x (zero x)~~

```
printf("z = %d\n", z); // 291
printf("z = %#o\n", z); // 443
printf("z = %#x\n", z); // 123
```

o/p : z = 123
z = 0173
z = 0x7b
y = 83
y = 0123
y = 0x53
z = 291
z = 0443
z = 0x123

* 111011001101011001001111111111 ⇒ 999999999

*
```
main()
{
 int y=0X1AB2;
 int z=0x1ab2;

 printf("y = %X\n", y);
 printf("z = %x\n", z);

}
```

o/p : y = 1AB2
z = 1ab2
} o/p depends upon printf format. i.e. on X and x.

when short & int same then long is of them &
when int & long same then short is half of them

size varies because size of int is 1 word

	Turbo C	GCC
short	2	2
int	2	4
long	4	4

```
* main()
{
    int x = 200 * 200 / 200;
    printf("%d", x);
}
```

signed -25536

o/p ⇒ -127

```
* main()
{
    long x = 200 * 200 / 200;
    printf("%d\n", x);
}
```

o/p ⇒ -127

```
* main()
{
    float test = 123.456;
    if (test = 666.777)
        printf("I am here");
    else
        printf("I am there");
}
```

value is assigned at both place

o/p ⇒ I am here.

In intermediate code machine alphabets are used.

Non-zero = True

Zero = false

Page No.
Date

* main()
{
float test=123.456;
if (test==666.777)
printf("I am here");
else
printf("I am there");
}

O/P : I am there.

* main()
{
float test =123.456;
printf ("%d bytes\n", sizeof(float));
printf ("%d bytes\n", size of (test));
printf ("%d bytes\n", size of (123.456));
printf ("%d bytes\n", size of (123.456f));
}

O/P : 4 bytes
4 bytes
8 bytes ⇒ by default all floating pt. const. are considered as double in C, C++ & Java.
4 bytes

* main()
{
float test=123.456;
printf("%e\n", test);
printf("%f\n", test);
printf("%g\n", test);
}

o/p: 123.4560e+02
123.456001
123.456

e=exponential
f=fixed format
g=significant digit (neces... digit

* main()
{
float test=543e-2f;
printf("%e\n", test);
printf("%f\n", test);
printf("%g\n", test);
}

o/p: 5.430000e+00
5.430000
5.43

* main()
{
long float test=123.45e-1; //long float=doubl
printf("%le\n", test);
printf("%lf\n", test);
printf("%lg\n", test);
}

o/p = 1.234500e+01
12.345000
12.345

double co

float co

Expr

i/o
dectar

getch
getche
getchar
scanf
gets

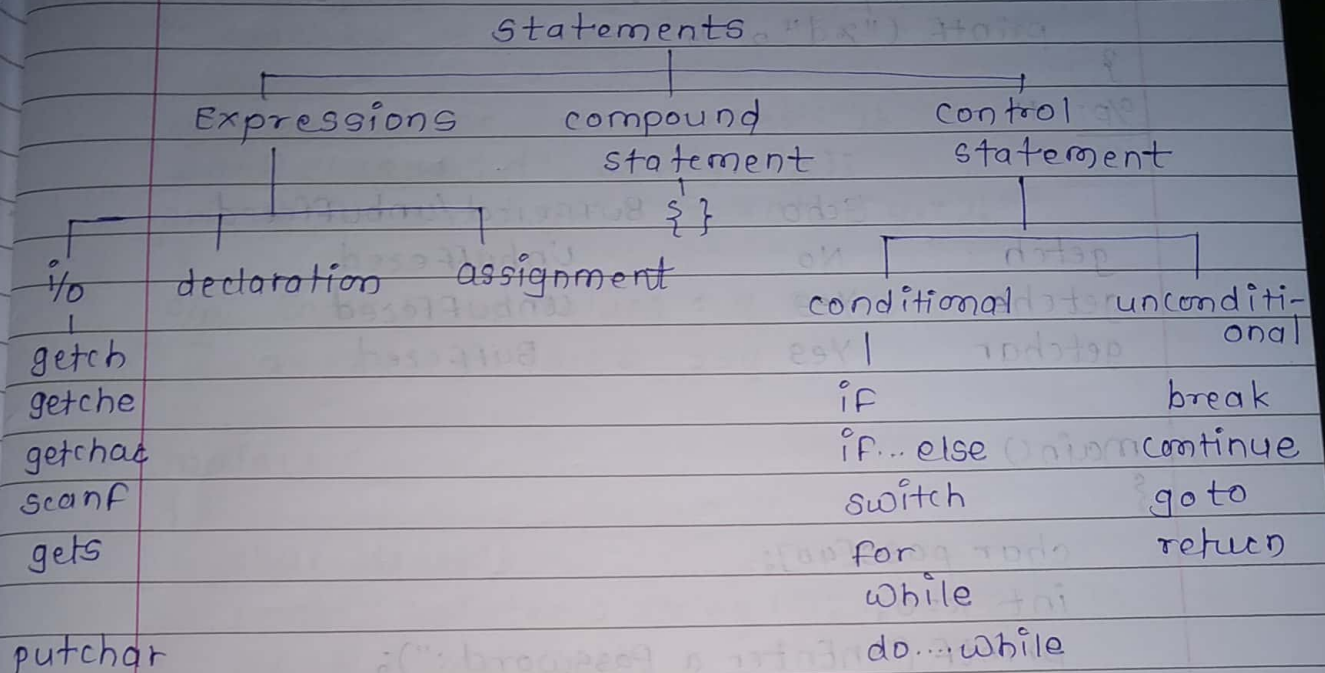
putchar
printf
puts

#

Macro

double constant ⇒ ex, 123.456, 0.0, 0., .0, 123.45e-2, 123e+1

float constant ⇒ ex. 123.456f, 0.0f, 0.F, .0f, 123.45e-2f, 123e+1f




```
#define abc 10
main()
{
   int x=abc;
   printf("abc %d xyz", x);
}
```

o/p: abc10xyz

Macro

```
#define square(x) x*x
main()
{
   int x=square(2+3);
   printf("%d",x);
}
```

2+3*2+3

o/p: 11

	Echo	Buffered/unbuffered
getch	No	Unbuffered
getche	Yes	unbuffered
getchar	Yes	Buffered

```
* main()
{
   char pass[40];
   int k=0;
   printf("\nEnter a Password:");
   do
   {
    pass[k++]=getchar();
    if(pass[k-1]==13)
    break;
    printf("\b");
    delay(100);
    putchar(42);
    pass[k-1]=0;
    printf("\n You Entered : %s",pass);
}
```

o/p: Enter the password: *******
You Entered: computer

Page No.
Date

* main()
{
int x;
float y;
char z;

printf("\nEnter Three Values :");
scanf("%c%f%d", &z, &y, &x);
printf("\n x=%d, y=%F, z=%c", x, y, z);
}

o/p: Enter Three Values : q 34.56 789
x=789, y=34.56, z=q

* main()
{
char str[30];
printf("\nEnter a string :");
scanf("%s", str);
printf("\n You Entered : %s ", str);
}

o/p: Enter a string : This is a test line
You Entered : This //Spaces are terminating

* main()
{
char str[30];
printf("\nEnter a String :");
gets(str);
printf("\n You Entered : %s", str);
}

o/p: Enter a String : This is a test line
You Entered : This is a test line

```
* main()
{
  char str[30];
  printf("\nEnter a string :");
  scanf("%[^g]", str); //Regular Expression
  printf("\nYou Entered :%s", str);
}
```

o/p:

```
Enter a string : abcdefg965
You Entered : abcdef
```

```
* main()
{
  char str[30];
  pf("\nEnter a Name :");
  scanf("%[a-zA-Z']", str);
  pf("\nYou Entered : %s", str);
}
```

o/p: Enter a Name : Allen D'Souza

```
You Entered : Allen D'Souza
```

```
* main()
{
  char str[30];
  gets(str);
  puts(str);
}
```

o/p: abcdefg

```
abcdefg
```

cursor → -

* main()
{
char str[30];
gets(str);
printf("%s",str);
}
o/p: abcdefg
abcdefg-← cursor

* ... printf format specifiers

In ... printf format strings, format specifiers have the following form

% [flags][width][.prec][F|N|h|l|L] type-char
Input size modifier

Each format specifier begins with the percent character(%).

* main()
{
int x=555, y=-555;
printf("abc%+dabc\n",x);
printf("abc%+dabc\n",y);
}
o/p: abc+555abc
abc-555abc
⊖ ← cursor

* main()
{
int real=33, imag=-44;
printf("%d%+di",real,imag);
}

if (%d+%d
then o/p=33+-44i

o/p: 33-44e

```
* main()
  {
     int test=1234;
     pf("abc%dabc\n", test);
     pf("abc%5dabc\n", test);
     pf("abc%+5dabc\n", test);
     pf("abc%6dabc\n", test);
     pf("abc%+6dabc\n", test);
     pf("abc%-6dabc\n", test);
     pf("abc%+-6dabc\n", test);
  }
```

o/p:
abc1234abc
abc-1234abc
abc+1234abc
abc 1234abc
abc +1234abc
abc1234 -abc
abc+1234 abc

* Blank flag: It gives space before printing of non negative no.

```
main()
{
  int x=44, y=-44, z=0;
  pf("abc% dabc\n", x);
  pf("abc% dabc\n", y);
  pf("abc% dabc\n", z);
}
```

o/p: abc 44abc
abc-44abc
abc 00abc

```
main()
{
 float x=123.45467f
 pf("x=%f\n",x);
 pf("x=%.2f\n",x);
 pf("x=%.1f\n",x);
 pf("x=%.0f\n",x);
 pf("x=%.#.0f\n",x);
}
```

o/p: 123.454697
123.45
123.45
123.
123.

```
* main()
 {
  int x=1234;
 pf("$$$ %d$$$\n",x);
 pf("$$$ %3d$$$\n",x);
 pf("$$$ %4d$$$\n",x);
 pf("$$$ %5d$$$\n",x);
 pf("$$$ %05d$$$\n",x);
 pf("$$$ %-5d$$$\n",x);
 pf("$$$ %+-5d$$$\n",x);

 pf("$$$ %-05d$$$\n",x);
 pf("$$$ %+05d$$$\n",x);
```

```
pf("$$$ %6d$$$\n",x);
pf("$$$%06d$$$\n",x);
pf("$$$ %+06d$$$\n",x);

pf("$$$ %-6d$$$\n",x);
pf("$$$%+6d$$$\n",x);
}
```

o/p:
```
$$$ 1234$$$
$$$1234$$$
$$$1234$$$
$$$01234$$$
$$$+1234$$$
$$$1234 $$$
$$$ 1234$$$
$$$  1234$$$
$$$ 01234$$$
$$$+1234$$$
$$$ 1234 $$$
$$$+1234 $$$
```

*
```
main()
{
  int t=333, z=-333;
  pf("$$$ %*d$$$\n",5,t);
  pf("$$$ %0*d$$$\n",5,t);
  pf("$$$ %*d$$$\n",5,z);
  pf("$$$%0*d$$$\n",5,z);
}
```

o/p:
```
$$$  333$$$
$$$00333$$$
$$$ -333$$$
$$$-0333$$$
```

```
main()
{
  printf("abcd\rpqrs\t$");
  printf("\nxyz");
}
```

o/p : pqrs $
xyz

```
main()
{
  printf("abcd\rpq\n$");
  printf("\nxyz");
}
```

o/p : pqcd
$
xyz

```
main()
{
  printf("abcd\rpq\t$");
  printf("\nxyz");
}
```

o/p : pq $
xyz

\n	newline(10) → ASCII
\t	horizontal tab(9)
\r	return to home(13)
\b	backspace (8)
\"	to print double coat
\\	to print \n

\P
\V
\a

```
main()
{
  printf("abcd\tpqrs\b\b xyz");
}
```

o/p : abcd pqxyz

```
main()
{
  printf("0\b-");
}
```

o/p : -

on paper o/p : Θ(theta)

```
main()
{
  printf("male/female\b\b\b\b\b\b------\b");
  printf("male/female\r-----");
}
```

o/p : male/------
----/female

o/p on paper : male/~~female~~
~~male~~/female

```
main()
{
  printf("\"abcd\"");
}
```

o/p : "abcd"

" " ⇒ control string

' ' ⇒ character const.

Page No.
Date

* main()
{
char x = '\'';
printf("%c abcd %c", x, x);
}
O/P: 'abcd'

* main()
{
printf("\\n");
}
O/P: \n

\ddd (octal)
\xuu (Hexadecimal)

* main()
{
printf("\052\n");
printf("\x2a");
}
O/P: *
*

ASCII of * = 42

0010 1010 ⇒ 42
000 101 010 ⇒ 052 (octal)
0010 1010 ⇒ 2a (Hexadecimal)

\0 (null - used in strings)

\ = To continue printf statement

```
main()
{
  printf("this is\
    a test line");
}
```

O/P : this is a test line

*
```
main()
{
  int y=100, x;
  x=printf("abcde%d", y);
  printf("\nx=%d", x);
}
```

O/P : abcde100
x=8

* printf returns integers. to given characters.

*
```
main()
{
  printf("%d", printf("abcde"));
}
```

O/P : abcde5

; = NULL statement.
* empty compound statement is equivalent to Null statement.

Page No.
Date

* main()
{
char y;
printf("\nThis is a test line");
int x; // declaration is not allowed here.
}
o/p: ERROR: Declaration not allowed after any statement.

* //scope or life of a variable

* main()
{
int x; // 'x' is declared in the outer compound statement
{
x=44;
printf("%d\n", x);
x=99;
}
printf("%d", x);
}
o/p: x=44
x=99

* main()
{
{
int x=44; // 'x' is declared in the inner compound statement
printf("%d\n", x);
x=99;
}

```
    printf("%d",x); //ERROR: undefined symbol 'x'.
}
```

*
```
main()
{
  {
    int x=44;
    printf("%d",x);
  }
}
```

o/p : 44

*
```
main()
{
  int x=99;
  {
    int x=44; //shadowing
    printf("x=%d, x=%d\n",x,x);
  }
}
```

o/p: x=44
x=44

Operators.

	Category	Operator	What it will (or does)	Associativity
1.	Highest	()	Function call	L→R
		[]	Array subscript	
		->	C++ Indirect component Selector	
		::	C++ scope access/ resolution	
		.	C++ direct component selector.	
2	Unary	!	logical negation (NOT)	R→L
		~	Bitwise (1's) complement	
		+	unary plus	
		-	unary minus	
		++	preincrement or postincrement	
		--	predecrement or postdecrement	
		&	address	
			Indirection	
		sizeof	(returns size of operand, in bytes)	
		new	(dynamically allocates c++ Storage)	
		delete	(dynamically deallocates C++ storage)	
		(type-caste)	Type caste operator.	
3.	multiplicative	*	multiply	L→R
		/	Divide	
		%	Remaindec(modulus)	

4.	Member access	.*	C++ dereference	L→R
		->*	C++ dereference	
5.	Additive	+	Binary plus	L→R
		-	Binary minus	
6.	Shift	<<	shift left	L→R
		>>	shift right	
7.	Relational	<	Less than	L→R
		<=	less than or equal to	
		>	greater than	
		>=	greater than or equal to	
8.	Equality	==	Equal to	L→R
		!=	Not equal to	
9.		&	Bitwise AND	
10.		^	Bitwise XOR	
11.		|	Bitwise OR	
12.		&&	Logical AND	
13.		||	logical OR	
14.	conditional	?:	(a?x:y means 'if a then x, else y'	R→L
15.	Assign-ment	=	simple assignment	R→L
		*=	Assign product	
		/=	Assign quotient	

= Associativity from R→L

	%=	Assign Remainder(modulus)
	+=	Assign sum
	-=	Assign difference
	&=	Assign bitwise AND
	^=	Assign bitwise XOR
	|=	Assign bitwise OR
	<<=	Assign left shift
	>>=	Assign right shift
16.	comma	, Eraluate.

R→L

R→L

~, <<, >>, &, ^, | = works on binary system

%d = signed decimal int

%u = unsigned decimal int

```
main()
{
  int x = (1,2,3,4,5);
  printf("%d", x);
}
```

o/p : 5

```
main()
{
  int x = (1,2,3,4,5)/(1,2);
      printf("%d", x);
}
```

o/p = 2

*
```
main()
{
  int x=33, y=33, z=0;
  printf("%d\n", !x);
  printf("%d\n", !y);
  printf("%d\n", !z);
}
```

o/p : 0
0
1

*
```
main()
{
  int x=-5, y;
      y=!x;
  printf("x=%d, y=%d", x, y);
}
```

o/p : x=-5
y=0

Every Non-zero value is True
'0' is false
} because in C there is no boolean data type.

* main()
{
printf("%d", ~0);
}
O/P : -1

int requires 16 bit, so in 16 bits write 0
i.e. all zero its negation is all 16 one's.
That no. is machine undecstandable, so, take 2's C
& ans. is -1.

* main()
{
printf("%d\n", ~0);
printf("%u", ~0);
}
O/P : -1
65535

* main()
{
printf("%d\n", ~3);
}

O/P : -4

0000 0000 0000 0011 ⇒ 3
1111 1111 1111 1100 ⇒ -3
0000 0000 0000 0100 ⇒ -4 ⇒ 2's C of (-3)

* main()
{
int x=5, y;
y=x++;
printf("x=%d, y=%d", x, y);
}
o/p: 6
5

* main()
{
int x=5, y;
y=++x;
printf("x=%d\n, y=%d", x, y);
}
o/p: x=6
y=6

* main()
{
int x=5, y;
y=x--;
printf("x=%d, y=%d", x, y);
}
o/p: x=4
y=5

* main()
{
int x=5, y;
y=-x;
printf("x=%d, y=%d", x, y);
}

```
main()
{
 int x=5, y;
 y= - -x;  // minus of minus
 printf("x=%d, y=%d", x, y);
}
```

o/p: x = 5, y=5

*
```
main()
{
 printf("%d bytes\n", sizeof(double));
 printf("%d bytes\n", sizeof(long));
 printf("%d bytes\n", sizeof(0));
 printf("%d bytes\n", sizeof(0.0));
 printf("%d bytes\n", sizeof(signed));
 printf("%d bytes\n", sizeof(unsigned));
 printf("%d bytes\n", sizeof(0L));
}
```

o/p : 8 bytes
4 bytes
2 bytes
8 bytes
2 bytes
2 bytes
4 bytes

*
```
main()
{
 printf("%d bytes\n", sizeof(33/22));
}
```

o/p : 2 bytes

* main()
{
float x = 33/2;
printf("%f", x);
}
o/p: 16.000000

* main()
{
int x=33, y=2;
float z = (float) x/y;
printf("%.2f", z);
}
Typecast operator.
o/p: 16.50

* main()
{
int x=44, y=0, z;
z = x/y;
printf("After Division, z=%d", z);
}
o/p: Divide error.

* main()
{
int x=13, y=7;
printf("%d\n", x%y);
printf("%d\n", x/y);
}
o/p: 6
1

* main()
{
int x=43, y=7;
printf("%d\n", y/x);
printf("%d\n", y%x);
}
o/p: 0
7

* main()
{
printf("%f", 33.33 % 11.0); //ERROR: Illegal use of floating pt.
}
o/p: ERROR = Mod operator doesn't works on floating point nos.

* #include<math.h>
main()
{
printf("%lf", fmod(33.33, 11.0));
}
o/p: 0.330000

* main()
{
printf("%d", 23<<4);
}
o/p = 368

* main()
{
printf("%d", 15<<15)
}
o/p = -32768

In right shift if MSB bit is 1 then the time of ~~left~~ right shift from left side & vice a versa. If one (1) put 1.

```
main()
{
printf("%d", 150>>4);
}
```

o/p: 00000000 1001 0110 ⇒150

0000 0000 0000 1001 ⇒150>>4

```
* main()
{
   int x=33, y=-44;
   printf("x<y: %d", x<y);
   printf("x<=y: %d", x<=y);
   printf("x>y: %d", x>y);
   printf("x>=y: %d", x>=y);
}
```

o/p: x<y = 0

x<=y: 0

x>y: 1

x>=y: 1

```
/*
```

A	B	A&B	A^B	A|B	!A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	0	1	0

```
*/
```

```
main()
{
  int x=33, y=44;
  int a, b, c;

  a = x & y;
  b = x ^ y;
  c = x | y;

  printf("x&y : %d\n", a);
  printf("x^y : %d\n", b);
  printf("x|y : %d\n", c);
}
```

O/p: x&y : 32
x^y : 13
x|y : 45

```
0010  0001 ⇒ 33
0010  1100 ⇒ 44
0010  1101 ⇒ x|y
```

56
64

x = 120 0111 1000
y = 40   0010 1000
         1   1

x&y : 40
x^y : 80
x|y : 120

In rightshift if MSB bit is 0 then put '0' at the time of ~~left~~ right shift from left side & vice a versa. If one (1) put 1.

```
main()
{
printf("%d", 150>>4);
}
```

o/p: 00000000 1001 0110 ⇒150

0000 0000 0000 1001 ⇒150>>4

```
* main()
{
   int x=33, y=-44;
   printf("x<y: %d", x<y);
   printf("x<=y: %d", x<=y);
   printf("x>y: %d", x>y);
   printf("x>=y: %d", x>=y);
}
```

o/p: x<y = 0

x<=y: 0

x>y: 1

x>=y: 1

```
/*
```

A	B	A&B	A^B	A|B	!A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	0	1	0

```
*/
```

```
main()
{
printf("%d\n", 2&1);   //0
printf("%d\n", 14&1);  //0
printf("%d\n", 26&1);  //0
printf("%d\n", 38&1);  //0
printf("%d\n", 50&1);  //0
printf("%d\n", 62&1);  //0
printf("%d\n", 74&1);  //0
printf("%d\n", 86&1);  //0
printf("%d\n", 98&1);  //0
printf("%d\n", 100&1); //0

printf("%d\n", 1&1);   //1
printf("%d\n", 13&1);  //1
printf("%d\n", 25&1);  //1
printf("%d\n", 37&1);  //1
printf("%d\n", 49&1);  //1
printf("%d\n", 61&1);  //1
printf("%d\n", 73&1);  //1
printf("%d\n", 85&1);  //1
printf("%d\n", 99&1);  //1
printf("%d\n", 101&1); //1
}
```

0000 0000 0000 0010 ⇒ 2
0000 0000 0000 0001 ⇒ 1

0000 0000 0000 1110 ⇒ 14
0000 0000 0000 0001 ⇒ 1

0000 0000 0001 1010 ⇒ 26
0000 0000 0000 0001 ⇒ 1

0000 0000 0010 0110 ⇒ 38
0000 0000 0000 0001

0000 0000 0011 0010 // 50
0000 0000 0000 0001

0000 0000 0011 1110 ⇒ 62
0000 0000 0000 0001

0000 0000 0100 1010 ⇒ 74
0000 0000 0000 0001

0000 0000 0101 0110 ⇒ 86
0000 0000 0000 0001

0000 0000 0110 0010 ⇒ 98
0000 0000 0000 0001

0000 0000 0110 0100 ⇒ 100
0000 0000 0000 0001

0000 0000 0000 0001 ⇒ 1
0000 0000 0000 0001

0000 0000 0000 1101 ⇒ 13
0000 0000 0000 0001
0001 1001 ⇒ 25

~~0000 0000 0000 1011~~
0000 0000 0000 0001

0000 0000 0010 0101 ⇒ 37
0000 0000 0000 0001
0011

~~0010~~ 0001 ⇒ 49
0000 0000 0000 0001

0000 0000 0011 1101 ⇒ 61
0000 0000 0000 0001

0000 0000 0100 1001 ⇒ 73
0000 0000 0000 0001
0101

~~0100~~ 0101 ⇒ 85
0000 0000 0000 0001

0000 0000 0110 0011 ⇒ 99
0000 0000 0000 0001

~~00~~ 0110 ~~0100~~ 0101 ⇒ 101
0000 0000 0000 0001

Swapping of two Nos.

```
* main()
  {
    int x=44, y=55;
    printf("Before swapping : x=%d, y=%d\n", x, y);

    temp=x;
    x=y;
    y=temp;

    printf("After swapping : x=%d, y=%d\n", x, y);

  }
```

O/p: Before swapping : x=44, y=55

After swapping : x=55, y=44

```
* main()
  {
    int x=44, y=55;
    printf("Before swapping : x=%d, y=%d\n", x, y);
```

```
x=x+y;          |  x=x*y          |  x=x^y;
y=x-y;          |  y=x/y          |  y=x^y;
x=x-y;          |  x=x/y          |  x=x^y;
```

```
    printf("After swapping : x=%d, y=%d\n", x, y);
  }
```

O/p: Before swapping : x = 44/55, y=55

After swapping : y=44, x=55

^ = carrate, XOR, circumplex.

gotoxy(); ⇒ takes cursor at (x,y) pt. on o/p screen.

Page No.
Date

0000 0000 0001 0111 ⇒ 23
0000 0000 0001 0010 ⇒ 18

* swapping of two nos. without using third variable in single statement.

```
main()
{
  int x=2, y=18;
  printf("Before swapping: x=%d, y=%d\n", x, y);

   y = (x+y)-(x=y);

   printf("After swapping = x=%d, y=%d\n", x, y);
}
```

o/p: Before swapping : x=2, y=18
After swapping : x=18, y=2

*
```
main()
{

  int x=33, y=22;
  printf("Before swapping: x=%d, y=%d\n", x, y);

  gotoxy(x,y);
  x=wherey();
  y=wherex();

  printf("After swapping : x=%d, y=%d\n", x, y);
}
```

o/p: Before swapping : x=22, y=33

```
main()
{
  int x=-1, y=0;
  if (x++ && ++y)
     printf("Here");
  else
     printf("\nThere");
  printf("x=%d, y=%d", x, y);
}
```

o/p : Here
x=0, y=1

```
main()
{
  int x=-1, y=-1;
  if(x++ && ++y)
     printf("Here");
  else
     printf("\nThere");
  printf("\n x=%d, y=%d", x, y);
}
```

o/p : There
x=0, y=0

```
main()
{
  int x=-1, y=0;
  if (++x && ++y)
     printf("Here");
  else
     printf("\nThere);
  printf("\n x=%d, y=%d", x, y);
}
```

o/p : There
x=0, y=0

```
main()
{
 int x=-1, y=-1;
 if (++x && ++y)
    printf ("Here);
 else
    printf ("\nThere);
 printf ("\n x=%d, y=%d," x, y);
}
```

o/p: There
x=0, y=-1

```
main()
 {
   int  x=-1, y=-1;
   if (++x  || ++y)
      printf ("\nHere");
   else
      printf ("\nThere");
   printf ("\nx=%d, y=%d", x, y);
}
```

o/p: There
x=0, y=0

```
main()
 {
  int x=-1, y=-1;
  if (++x || y++)
     printf ("Here");
```

```
else
  printf("\n There");
printf("\n x=%d, y=%d", x, y);
}
```

O/p: Here
x = 0, y = 0

*
```
main()
{
  int x=-1, y=-1;
  if (x++ || ++y)
    printf("Here");
  else
    printf("\n there);
  printf("\n x=%d, y=%d", x, y);
}
```

O/p: Here
x = 0, y = -1

*
```
main()
{
  int x = -1, y = -1;
  if (x++ || y++)
    printf("Here");
  else
    printf("\n there);
  printf("\n x=%d, y=%d", x, y);
}
```

O/p: Here
x = 0, y = -1

```
main()
{
  int x=44;
  if (x&1)
    printf("odd");
  else
    printf("Even");
}
```

o/p: Even

```
main()
{
  int x=4;
  x&1 ? printf("odd") : printf("Even");
}
```

o/p: Even

```
main()
{
  int x, a=-1;

  // a ? x=2 : x=3;  ERROR: Lvalue required
   x=a ? 2 : 3;
  printf("x=%d", x);
}
```

o/p : x=2

```
main()
{
  int x=2;
  x+=3 ; // x=x+3;
  x-=1;
  x*=4;
  x/=2;
  x%=9;
  x<<=2;
  x>>=1;
  x&=15;
  x^=15;
  x|=15;
  printf("\n x=%d", x);
}
```

o/p: x = 5

x = 4

x = 16

x = 8

x = 8

x = 32

x = 16

x = 0

x = 15

x = 15 ⇒ final o/p

0000 0000 0000 0010 ⇒ 2

0000 0000 0000 1111 ⇒ 15

0000 0000 0000 0010 ⇒ 2

0000 0000 0000 1101 ⇒ 13

0000 0000 0000 1111 ⇒ 15

```
main()
{
  printf("\nHii"), printf("\n Hello");
}
```

o/p : Hii
Hello

```
main()
{
  int x=5;
  printf("%d %d %d %d %d", x++, ++x, x++, ++x, x++);
}
```

push on stack ① ② ③ ④ ⑤

o/p: 9 9 7 7 5
(Stack operation is used)

```
main()
{
   int x=7;
   printf("%d %d %d %d %d %d %d", --x, x++, x++, --x,
                                  ++x, x++);
}
```

++x

o/p: 10 10 9 9 10 8 8

```
main()
{
   int a=3, b=4, c=5, d=6, z;
   z = a+b*c-d;
   printf("%d", z);
}
```

④ ② ① ③

o/p = 17

* main()
{
int a=33, b=44, c=55, d=11, z;
z = a | b-c/d^b%a*c&d<<d/2+(a+b);
printf("\n z=%d", z);
}

o/p:

z = a | b-c/d^b%a*c&d<<d/2+(a+b)

z = a | b-c/d^b%a*c&d<<d/2+77

z = a | b-5^b%a*c&d<<d/2+77

z = a | b-5^11*c&d<<d/2+77

z = a | b-5^11*55&d<<d/2+77

z = a | b-5^605&d<<d/2+77

z = a | b-5^605&d<<5+77

z = a | 39^605&d<<82

z = a | 39^605&0

z = a | 39^0

z = a | 39

z = 39

```
main()
{
   int a=44, b=33, c=55, d=11, z;
   z=a^b-c*d|b%a*c&d>>d%2+(a-b-6);
   printf("\nz=%d",z);
}
```

O/p: z=

z = a^b-c*d|b%a*c&d>>d%2+5

33
55
165
1650
1815

z = a^b-605|b%a*c&d>>d%2+5

z = a^b-605|33*c&d>>d%2+5

z = a^b-605|1815&d>>d%2+5

605
33
572

z = a^b-605|1815&d>>1+5

z = a^(-572)|1815&d>>1+5

z = a^(-572)|1815&d>>6

0000 1011

z = a^(-572)|1815&0

z = a^(-572)|0

z = -536|0

0000 0010 0011 1100 ⇒572

1111 1101 1100 0100 ⇒-572

0000 0000 0010 1100 ⇒44

1111 1101 1110 1000 ⇒-572+44

z=-536

```
main()
{
  int test1=3, test2=2, test3=5;
  test3 += test1 == test2 <= test1 != test2 >= test1 ? *2 : test2/2;
  printf("%d\n", test3);
}
```

o/p: 6

test3 += test1 == test2 <= test1 != test2 >= test1 ? *2 : test2/2;

test3 += test1 == test2 <= test1 != test2 >= test1 ? 6 : 1;

test3 += test1 == test2 <= test1 != test2 >= test1 ? 6 : 1;

test3 += test1 == 1 != test2 >= test1 ? 6 : 1;

test3 += test1 == 1 != 0 ? 6 : 1;

test3 += test1 == 1 != 0 ? 6 : 1;

test3 += 0 != 0 ? 6 : 1;

test3 += 0 ? 6 : 1;

test3 += 1;

test3 = 6

ex. Write a program to add two nos.

(i) with '+' operator

(ii) without '+' operator.

with '+' operator:

```
main()
{
    int x=5, y=4;
    printf("x+y=%d", x+y);
}
```

* if (keyword)
  conditional

Syntax:

- if (<expression>) <statement1>;
- if (<expression>) <statement1>;
  else <statement2>;

If <expression> is non-zero when evaluated, <statement1> is executed.

In the second case, <statement2> is executed, if the expression is 0.

An optional else can follow an if statement. but no statements can come betn an if statement and an else.

Ex.

```
if (count < 50) count++;
if (x<y)
    z= x;
else
    z=y;
```

* main()
{
int x=45;
if (x & 1)
printf ("Odd");;
else
printf ("Even");
}
ERROR: misplaced else

* main()
{
int x=45;
if(x & 1)
{
printf("Odd");
printf ("This");
}
else
printf("Even");
printf ("that");
}
O/P: OddThisthat

* find Greatest no.
main()
{
int x, y, z;
printf("Enter three distinct nos.");
Scanf ("%d %d %d", &x, &y, &z);

```
if(x>y && x>z)
    printf("%d is greatest",x);
else if (y>z)
    printf("%d is greatest",y);
else
    printf("%d is greatest",z);
}
```

*
```
main()
{
    int x=55, y=44, z=44;
    /*
        Buy 1 get 2 free
    */
    printf("\n You have to Pay:");
    if (x>y && x>z)
        printf("%d",x);
    else if (y>z)
        printf("%d",y);
    else
        printf("%d",z);
}
```

o/p: You have to pay : 55

*
```
main()
{
    int x=11, y=22, z=33;
    /*
        Buy 2 get 1 free
    */
```

```
printf("you have to pay :");
if (x>y && x>z)
{
    if (y>z)
        printf("%d %d", x+y);
    else
        printf("%d %d", x, z);
}
if (y>z)
{
if (x<y && x<z)
    printf("%d", x+y+z);
else if (y<z)
    printf("%d", x+z);
else
    printf("%d", x+y);
}
```

```
* main()
{
    int x=71, y=72, z=66;
    /*
      Buy 2 get 1 free
    */
    printf("You have to pay Rs. ");
    if(x>y && x>z
    {
        if (y>z)
            printf("%d", x+y);
        else
            printf("%d", x+z);
    }
```

```
else if (y > z)
 {
    if (x > z)
        printf("%d", y+x);
    else
        printf ("%d", y+z);
 }
 else
   {
      if (x > y)
          printf ("%d", z+x);
      else
          printf("%d", z+y);
   }
}
```

o/p: You have to pay Rs. 149

ques printing Ascending & descending seq. of nos

```
main()
{
    int x=22, y=99, z=35;

    if (x > y && x > z)   // 'x' is greatest
     {
        if(y > z)          // 'y' is middle one
        {
            Ascending:
          printf("%d %d %d \n", z, y, x);
          printf ("Desending: %d %d %d", x, y, z);
        }
        else    // 'z' is middle one
```

```
    {
      printf("Ascending : %d %d %d", y, z, x);
      printf("Descending : %d %d %d", x, z, y);
    }
}

else if (y > z)
{
    if (x > z)
    {
      printf("Ascending : %d %d %d", z, x, y);
      printf("Descending : %d %d %d", y, x, z);
    }
    else
    {
      printf("Ascending : %d %d %d", x, z, y);
      printf("Descending : %d %d %d", y, z, x);
    }
}

else
{
    if (x > y)
    {
      printf("Ascending : %d %d %d", y, z, x);
      printf("Descending : %d %d %d", x, z, y);
    }
    else
    {
      printf("Ascending : %d %d %d", x, y, z);
      printf("Descending : %d %d %d", z, y, x);
    }
}
}
```

O/p: As
De

que. Write

```
main
{
```

2016 = leap
1900 = not leap

Date

O/P: Ascending : 22, 35, 99
Descending : 99, 35, 22

Que. Write a program to check if a year is leap or not.

```
main()
{
    int x;
    printf("Enter year : ");
    scanf("%d", &x);

    if (x % 4 == 0 && x % 100 == 0)
    {
        printf("Year %d is not leap", x);
    }
    else
        printf
```

* main()

```
{
   char c;
   printf("\nEnter a character:");
   c=getchar();

   if (c=='r' || c=='R')
        printf("\n Red");
else if (c=='w' || c=='W')
        printf("\n White");
else if (c=='B' || c=='b')
        printf("\n Black");
else if (c=='p' || c=='P')
        printf("\n Pink");
else if (c=='o' || c=='O')
        printf("\n Orange");
   else
   pf("\nInvalid");
}
```

O/p:

Enter a character : R

Red

* main()

```
{
   char c;
   printf("\nEnter a small character : ");
   c=getchar();

   switch(c) //Integral expression
   {
    case 'r': pf("\n Red");
              break;
```

switch

Rules

switch, while, do...while, for ⇒ contains break.

Page No.
Date

```
case 'w':
        printf("White");
        break;
case 'g':
        printf("Green");
        break;
case 'b':
        printf("\nBlack");
        break;
case 'p': printf("\nPink");
        break;
default:
        printf("\nInvalid");
}
```

o/p: Enter a small character : p
Pink.

* switch contains only integral expression

Rules of case = mandatory
const expression
int type
unique

*
```
main()
{
  char c;
  printf("\nEnter a small character");
  c=getchar();
```

switch, while, do...while, for ⇒ contains break.

Page No.
Date

```
case 'w':
        printf("White");
        break;
case 'g':
        printf("Green");
        break;
case 'b':
        printf("\nBlack");
        break;
case 'p': printf("\nPink");
        break;
default:
        printf("\nInvalid");
}
```

o/p: Enter a small character : p
Pink.

* switch contains only integral expression

Rules of case = mandatory
const expression
int type
unique

*
```
main()
{
  char c;
  printf("\nEnter a small character");
  c=getchar();
```

```
switch(c) //integral expression
{
    case 'r':
    case 'R':
            printf("\nRed");
            break;
    case 'w':
    case 'W':
            printf("\nwhite");
            break;
    case 'g':
    case 'G':
            printf("\nGreen");
            break;
    case 'p':
    case 'P':
            printf("\nPink");
            break;
    case 'b':
    case 'B':
            printf("\nBlack");
            break;
    default:
            printf("\nInvalid");
}
printf("\nAftec switch");
}
```

O/p: Enter a character: G
Green
Aftec switch

* switch, case and default (keywords)

Branches control

Syntax:

- switch (<expression>) <statement>
- case <constant expression>:
- default:

switch

Causes control to branch to one of a list of possible statements in the block defined by <statement>

The branched-to statement is determined by evaluating <expression>, which must return an integral type

case

The list of possible branch points within <statement> is determined by preceding substatements with

case <constant expression>:

where <const-expression> must be an int and must be unique

Once a value is computed for <expression>, the list of possible <constant expression> values determined from all case statements is searched for a match.

If match is ~~not~~ found, execution continues after the matching case statement & continues until a break statement is encountered or the end of <statement> is encountered/reached.

default:

If a match not found and the "default:" statement prefix is found within <statement>, execution continues at this point.

Otherwise, <statement> is skipped entirely.

Ex: (enum)

```
switch (operand)
{                                  ← Symbolic consts.
    case MULTIPLY : x *= y; break;
    case DIVIDE   : x /= y; break;
    case ADD      : x += y; break;
    case SUBTRACT : x -= y; break;
    case INCREMENT1: x++;
    case ~~DE~~INCREMENT1: x++; break;
    case EXPONENT :
    case ROOT :
    case MOD:  printf("Not done\n"); break;
    default :  printf ("Bug!\n");

    exit(1);
}
```

* main(
{

```
* main()
 {
    int x;
    printf("\nEnter a digit :");
    scanf("%1d", &x);
    switch(x)
    {
      case 0:
            printf("zero");
            break;
      case 1:
            printf("One");
            break;
      case 2:
            printf("Two");
            break;

      case 3:
            printf("Three");
            break;
      case 4:
            printf("foue");
            break;
      case 5:
            printf("Fire");
            break;
      case 6:
            printf("Six");
            break;
      case 7:
            printf("Seven");
            break;
```

```
        case 8:
                printf("Eight");
                break;
        case 9:
                printf("Nine");
                break;
        default:
                printf("\nInvalid");
        }
}
```

*
```
main()
{
    float x=128.456f;
    switch(x)  //ERROR
    }
    }
}
```

ERROR: switch selection expression must be of integral type.

*
```
main()
{
    int x=1;
    int y=0;
    switch(x)
    {
     case 2:
            pf("yes");
            break;
```

```
    case y+1 : //ERROR
               pf("NO");
               break;
  }
}
```

ERROR: constant expression Required.

*
```
main()
{
  int x=5;
  switch (x++ &g)
  {
    default:
            printf("something\n");
    case 4:
            printf("this");
    case 2--1:
            printf("is");
    case 1:
            printf("a test");
    case 5:
            printf("line");
  }
}
```

O/p: a test line.

* main()
{
int i;
for(i=1 ; i=5 ; i++)
pf("%d\n", i, i);
}

o/p: 1
2
3
4
5

precision ⇒ Type → : n & .*

float test = 123.456703;
pf("%F\n", test);

* main()
{
for(printf("1\t"); printf("2\t"); printf("3\t"))
printf("4\t");
}

o/p: 1 2 4 3 2 4 3 2 4 3
2 4 3 2 4 3 2 4 3 2
4 3 2 -

* main()
{
int i;
for (i=1; i<=10; i++)
printf("%d\t", i);
printf("\n i=%d", i);
}

O/p: 1 2 3 4 5 6 7 8 9 10
i=11

For (keyword)

for loop

Syntax : for([expr1] ; [expr2] ; [expr3]) <statement>

<statement> is executed repeatedly UNTIL the value of <expr2> is 0.

BEFORE the 1st iteration, <expr1> is evaluated. This is usually used to initialize variables for the loop.

AFTER each iteration of the loop, <expr3> is evaluated. This is usually used to increment a loop counter.

In C++, <expr1> can be an expression or a declaration.

The scope of any declared identifier extends to the end of the control statement only.

All the expressions are optional. If <expr2> is left out, it is assumed to be 1.

Ex:

```
for(i=0 ; i<100; i++)
    sum+=a[i];

for(i=0; t=string ; i<40&& *t; i++,t++)
 putch(*t);
 putch('\n');
```

```
* main()
 {
    int i=1;
    for(;i<=100;)
        printf("%d\t",i++);
 }
```

o/p: 1 to 100

```
* main()
 {
    int i;
    for(i=1;i<=100;i++)
    {
        printf("%d",i);
        if(i%1)
            printf(".");
        printf("\t");
    }
}
```

O/p: 1. 2 3. 4 5. 6 7. 8 9. 10
11. 12 13. 14 15. 16 17. 18 19. 20
. . . .
. . . .
91. 92 93. 94 95. 96 97. 98 99. 100

```
* main()
  {
    int i;
    for (i=1; i<10; ++i);
        printf("%d\t", i);
    printf("\ni=%d", i);
  }
```

O/p: 10
i=10

```
* main()
  {
    int i;
    for(i=1; i<=100; i++)
    {
      if (i%3)  // Not divisible by 3
          printf("%d", i);
      printf("\t");
    }
  }
```

O/p: 1 2 4 5 7 8

```
main()
{
    int i;
    for(i=1; i<=100; i++)
        if(!(i%3) && i%5)   // divisible by 3 & not by 5
            printf("%d\t", i);
}
```

O/p:

3	6	9	12	18	21	24	27
33	36	39	42	48	51	54	57
63	66	69	72	78	81	84	87
93	96	99					

```
main()
{
    int i;
    for(i=1; i<=100; i++)
        if(i%3 || i%5)
            pf("%d\t", i);
}
```

O/p:

1	2	3	4	5	7	8	9	10	11	12	13	14	16
17	18	~~19~~	19	20	-	-	-	-	-	-	-	-	-

```
main()
{
    int i;
    for(i=1; i<=100; i++)
        if(i%3 || !(i%5) && i%7)
            printf("%d\t", i);
}
```

divisible by 3 but not 5

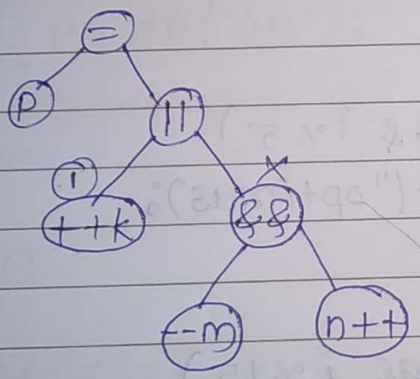
According to expression tree.

O/p: 1 2 3 4 5 7 8 10 11 13
14 15 16

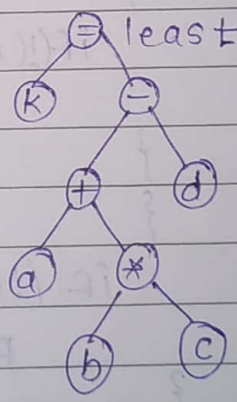
* main()
{
int k=0, m=1, n=0, p;
p = ++k || --m && n++;
printf("k=%d, m=%d, n=%d, p=%d", k, m, n, p);
}

O/p: p=1, k=1, m=1, n=0

p = ++k || --m && n++



k = a+b*c-d



least priority.

1 2 optimus 4 prime
prime 11 optimus 13 14 optimus-p
optimus 19 prime optimus 22 23 optimus prime

```
main
{
int i;
for(i=1; i<=100; i++)
    if((i%3) || !(i%5))
        printf("optimus");
```

```
int main()
{
  int i;
  for(i=1; i<=100; i++)
  {
    if(i%3 && i%5 && i%15)
      printf("%d\n", i);
      {
        if (!(i%3) && !(i%5))
          printf("optimus-prime\t");
      }
      }
      if(!(i%3) && i%5)
        printf("optimus");
      }
      {
      if (!(i%5) && i%15)
        printf("prime\t");
    }
  }
}
```

O/P:

```
* main()
{
    int j, i;
    for (i=1; i<=5; i++)
    {
        for (j=1; j<=5; j++)
            printf("%d", j);
        /*
            j=1;
            printf("%d", j++);
            printf("%d", j++);
            printf("%d", j++);
            printf("%d", j++);
            printf("%d", j++);
        */
        printf("\n");
    }
}
```

```
* main()
{
    int i, j;
    for (i=1; i<=5; i++)
    {
        for (j=1; j<=5; j++)
            printf("%d", j);
        printf("\n");
    }
}
```

O/p:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
```

```
main()
{ int i,j;
  for(i=1;i<=5;i++)
    {
      for(j=1;j<=i;j++)
        printf("%d",6-j);
      printf("\n");
    }
}
```

i	j
x	y
1	1
2	2
3	3
4	4
5	5

o/p:
```
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

```
main()
{
  int i,j;
  for(i=1;i<=5;i++)
  {
    for(j=1;j<=6-i;j++)
      printf("%d",6-j);
    printf("\n");
  }
}
```

o/p:
```
5 4 3 2 1
5 4 3 2
5 4 3
5 4
5
```

x	y
1	5
2	4
3	3
4	2
	1

```
main()
{
   int i,j;
   for(i=1;i<=5;i++)
   {
     for(j=1;j<=10-2*i;j++)
          printf(" ");
     for(j=1;j<=i;j++)
          printf("%d",j);
     printf("\n");
   }
}
```

o/p:

```
         1
       1 2
     1 2 3
   1 2 3 4
 1 2 3 4 5
```

```
* main()
{
  int i,j;
  for(i=1;i<=5;i++)
   {
     For(j=1;j<=10-2*i;j++)
          printf(" ");
     for(j=1;j<=i;j++)
          printf("%d ",i+1-j);
     printf("\n");
   }
}
```

```
1
2 1
3 2 1
4 3 2 1
```

x y
1 8
2 6

y=8

2x-2

2(x-1)

y=2(1-1)

y=x=9-

10-2(1)=8

10-2(2)=6

10-2(3)=4

```
main()
{
    int i,j;
    for(i=1; i<=5; i++)
    {
        for(j=1;j<=10-2*i ;j++)
            printf(" ");
        for (j=i; j>=1;j--);
            printf("%d ",j);
            printf("\n");
    }
}
```

Op:

```
        1
      2 1
    3 2 1
  4 3 2 1
5 4 3 2 1
```

```
main()
{
    int i,j;
    for(i=5 ; i>=1 ; i--)
    {
      for(j=1 ; j<= 2*i-2 ; j++)
```

```
main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1;j<=5-i;j++)
            printf(" ");
        for(j=1;j<=i;j++)
            printf("* ");
        printf("\n");
    }
}
```

~~for~~ o/p:

```
    * 
   * * 
  * * * 
 * * * * 
* * * * *
```

```
main()
{
  int i,j,n;
  printf("\nEnter ODD no. of rows:");
  scanf("%d",&n);

  if(!(n&1))
  {
    printf("\nAre You mad ???");
    exit(1);
  }
```

```
for(i=1;i<=n/2+1;i++)
{
  for(j=1;j<=n/2+1-i;j++)
    printf(" ");
  for(j=1;j<=i;j++)
    printf("* ");
  printf("\n");
}

for(i=1;i<=n/2;i++)
{
  for(j=1;j<=i;j++)
    printf(" ");
  for(j=1;j<=n/2+1-i;j++)
    printf("* ");
  printf("\n");
}
}
```

O/P: Enter ODD no. of rows : 9

```
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
```

Static block
Static member fu?
static data members/variables
static fu

This → ptr. o/p to no

(*) o/p:

```
         *
       * * *
     * * * * *
   * * * * * * *
 * * * * * * * * *
```

```
main()
{
    int i, j;
    for(i=1; i<=5; i++)
    {
        for(j=1; j<=10-2*i; j++)
            printf(" ");
        for(j=1; j<=2*i-1; j++)
            printf("* ");
        printf("\n");
    }
}
```

i	2*i-1
1	1
2	3
3	5
4	7
5	9

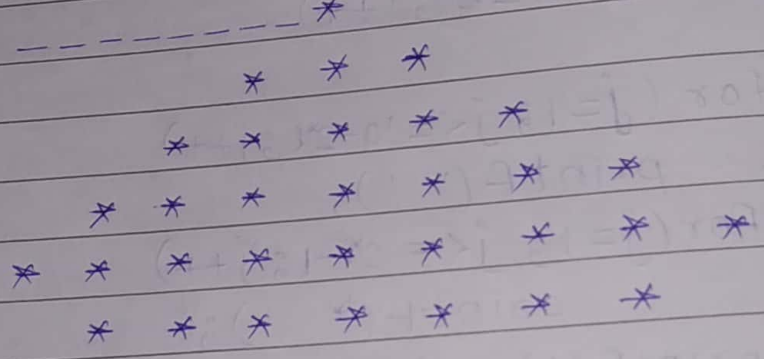
2-1
4-1
6-

j=1; j<=

```
main()
{
  int i, j;
  for(i=1; i<=5; i++)
  {
    for(j=1; j<=10-2*i; j++)
        printf(" ");
    for(j=1; j<=2*i-1; j++)
        printf("* ");
    printf("\n");
  }
}
```

```
for(i=1;i<5;i++)
{
    for(j=1;j<=2i;j++)
        printf(" ");
    for(j=1;j<=9-2i;j++)
        printf("* ");
    printf("\n");
}
}
```

o/p:



```
* main()
{ int i;
  for(i=1;i<=10;i++)
  {
    printf("%d",i);
    if(i%1 && !(i%5))
      break; //abnormal termination
  }
printf("\n After for(;;), i=%d", i);
}
```

// If termina
occurs with
it is norma
terminati

o/p:

1 2 3 4 5

After for(;;) i=

F7=Trace

Page No.
Date

* main()
{
int i;
for(i=1; i<=10 ; i++)
{
if(i&1) && !(i%5))
break;
printf("%d", i);
}
printf("\nAftee for(;;), i=%d", i);
}

o/p: 1 2 3 4
Aftee for(;;), i=5

* main()
{
int i;
for (i=1 ; i<=20; i++)
{
printf("%d", i);
if (i & 1)
{
printf(" .. ");
continue; // jump to i++
}
printf("\n");
}
printf ("\nAftee for(;;), i=%d", i);
}

o/p: 1..2
3..4
5..6
7..8
9..10
11..12
13..14
15..16
17..18
19..20
After for(;;), i=2|

* main()
{
int i;
for(i=1; i<=10; i++)
{
if (i&1)
continue;
printf("%d", i);
}
}

o/p: 2 4 6 8

* main()
{
printf("1\t");
while (printf("2\t"))
{
printf("4\t");
printf("3\t");
}
}

O/p: 1 2 4 3 2 4 3 2 4 3 2 4 3
2 4 3 2 4 3 _ _ _ _

* main()
{
int sum=0;
unsigned long num;
printf("\nEnter the no:");
scanf("%lu", &num);

while(num)
{
sum+=num%10;
num/=10;
}
printf("~~%d~~ Sum: %d", sum);
}

O/p:
Enter the no.: 12345

Sum=15

* write a program to reverse a no.
{
int sum=0~~, no~~;
unsigned long num~~, no~~=0;
printf("\nEnter the no.:");
scanf("%lu", &num);

while(num)
{
~~num/=10;~~

```
    no = num % 10;
    printf("%d", no);
    //sum+=no;
    num/=10;
  }
  printf("NO: %ln", no);
  printf("Sum: %d", sum);
}
```

* main()

```
{
  int sum=0;
  unsigned long num, sum=0UL;
  printf("\nEnter no.");
  scanf("%lu", &num);

  while(num)
  {
    sum+=(num%10)*10;
    sum+=10+num%10;
    num/=10;
  }
  printf("Rerersed NO: %lu", sum);
}
```

* Write a program to check if a no. is palindrom or not.

→ main()

```
{
  unsigned long num;

  printf("\nEnter NO:");
  Scanf("%lu", &num);
```

* take

main

{

}

```
while(num)
{
    sum=sum*10+num%10;
    num/=10;
}
if (sum==no)
  {
    printf("No. is palindrome");
  }
else
    printf("No. is not palindrome");
```

* take no. as i/p & check how many 2 in that no.

```
main()
{
    unsigned long num=1,sum;
    int count=0, sum=0;
    printf("\nEnter no:");
    scanf("%lu",&num);
    while(num)
    {
        sum=num%10;
        if (sum==2)
        {
            ++count;
        }
        num/=10;
    }
    printf("No. of 2 :%d", count);
}
```

* While (keyword)

Repeats execution

Syntax : while (<expression>) <statement>

<statement> is executed repeatedly as long as the value of <expression> remains non-zero

The test takes place before each execution of the <statement>.

Example: while (*p==' ') p++;

```
main()
{
   unsigned long num;
   int count=0;
   printf("\nEnter a NO :");
   scanf ("%lu", &num);
   printf ("%lu ",num);
   while (num)
   {
     if (num%10==2)
        count++;
     num/=10;
   }
   printf (" contains %d 2s ", count);
}
```

How many nos divisible by 3 in 1 to 1000

→

```
main()
{
    int count=0;
    int i;
//printf("\nEnter a Number:");
// scanf("%lu", &num);

    while{
    for(i=1; i<=1000; i++)
    {
        if(!(i%3))
            count++;
    }
printf("\n Numbers divisible by 3 : %d", count);
}
```

```
*main()
{
   int
```

* Check No. is Prime or Not.

① →
```
int main()
{
  int num, i;
  printf("Enter a No:");
  scanf("%d", &num);

  for(i=2; i<=num/2 ; i++)
  {
    if (!(num%i))
    {
      printf("\n Not prime");
      exit(0);
    }
  }
  printf("\n No. is prime");
}
```

② →
```
main()
{
  int num, i; flag=0;
  printf("Enter a No");
  scanf("%d", &num);
  for(i=2; i<=num/2 ; i++)
  {
    if (!num%i)
    {
      flag=1;
      break;
    }
  }

  if (flag)
    printf("\n Not Prime");
  else printf("\n Prime");
}
```

Page No.
Date

(*) Find no. is perfect no. or not.

```
main()
{
    int num, i, sum=0;
    printf("\nEnter a Number:");
    scanf("%d", &num);

    for(i=1; i<=num/2 ; i++)
    {
        if (!(num%i))
        {
            sum+=i;
        }
    }
    if (sum==num)
    printf("\n No. is perfect No");
    else
        printf("\n No. is not perfect No.");
}
```

num=5

i = (num-1) 4-1 3

(*) ~~find~~ Program for strong no (145)
1! + 4! + 5! = 145

4×3×2×1

```
main()
{
 int num, sum1=0, sum=0, no=1, i;
     printf("\nEnter NO");
     scanf("%d", &num);

     while(num)
     {
         Sum=num%10;
```

sum=1; 2

for(i=1; i<=num; i++)
{
if(sum=sum*i;
}

sum=1*i;

1=

```
for(i=sum; i<=sum; i--)
{
    no = no * i;
}
sum1=sum1+no;
num = num/10;
}
if(sum == num) strong
    printf("\n Perfect no");
else
    printf("\n Not a perfect strong no.");
}

main
{
  int num, sum=0, sum1=0, no=1, i;
  printf("\nEnter No:");
  scanf("%d", &num);

  while(num)
  {
     sum = num % 10;
     for(i=sum
```

```
main()
{
    int num, no1, num1=0, sum=0, no=1, i;
    printf("\nEnter No:");
    scanf("%d", &num);
     no1 = num;
    while (num)
    {
        num1 = num%10;

        for (i=num1, no=1; i>=1; i--)
        {
            no = no * i;
        }
        sum = sum+no;
        num = num/10;
    }

    if (sum == no1)
        printf("\n Strong NO");
    else
        printf("\n Not a Strong no");
}
```

Armstrong No: 371, 153, 370, 407

$371 = 3^3 + 7^3 + 1^3 = 371$

8208, 1634

$8^4 + 2^4 + 8^4 = 8208$

* Program for Armstrong No.

```
main()
{
  int num, no, i, count=0, sum=0, mod, y=1;
  printf("\nEnter No :");
  scanf("%d", &num);
  no=num;

  while(num)
  {
    num/=10;
    count++;
  }
  num=no;
  while(num)
  {
    mod=num%10;
    for(i=1; i<=count; i++)
    {
      y*=mod;
    }
    sum+=y;
    num/=10;
  }
  if(sum==no)
    printf("Armstrong No");
  else
    printf("\n Not Armstrong No");
}
```

* Display Armstrong No. orderwise.

1 = 1

5

37

8208

* No divisible by 3 without multiplicative operator

```
main()
{
   int num;
   printf("\n Enter a Number :");
   scanf("%d", &num);

   if(num >0)
   {   while (num>0)
   while (num)==0)
    {
      num = num-3;
    }
   }
   else
   {
    while (num)<0)
    {
      num = num+3;
    }
   }

 if (num==0)
   printf("\n No. is divisible by 3");
 else
    printf("\n No. is not divisible by 3");

}
```

leap year
if(!(year%400) || (year%4) && year%100)

```
main()
{
    int num;
    printf("\nEnter a No:");
    scanf("%d", &num);

    num*=num<0?-1:1;    //OR num*=num;
                        //OR num*=pow(-1,num>>15&1);
                                  (int)
    while(num>0)
        num-=3;

    if(!num)
        printf("\n Divisible");
    else
        printf("\n Not divisible");
}
```

```
main()
{
   int x=14;
   while (x--)
      printf("This is a test line: %d\n", --x);
}
```

o/p:

```
This is a test line: 12
This is a test line: 10
This is a test line: 8
This is a test line: 6
This is a test line: 4
This is a test line: 2
This is a test line: 0
```

```
main()
{
   int x=0;
   while (x);
      printf("This is a test line: %d\n", x--);
   printf("\n x=%d", x);
}
```

o/p: This is a test line : 0

x = -1

```
main()
{
  int x=0
  do
  {
     printf("This is a test line: %d\n", x--
  }while(x);
```

```
    printf("\n x=%d", x);
}
```

O/p: This is a test line: 0
: -32768
: 32767
This is a test line: 1
x=0

O/p: This is a test line: 0
This is a test line: -1
This is a test line: -32768
This is a test line: 32767
This is a test line: 1
x=0

* main()
```
{
  do
  {
   printf("This is a test line");
  } while(0);
}
```

O/p: This is a test line.

do .... while loop

Do.... while loop

syntax: do<statement> while (<expression>);

<statement is executed repeatedly as long as the value of <expression> remains non-zero.

The test takes place AFTER each execution of the <statement>

Example:

```
i=1 ; n=1
do{
    n*=i;
    i++;
}while (i<=factorial);
```

```
* main()
{
  int x=1;
  do
  {
    if (x%3)
        continue;
    printf("%d\t", x%1);
  } while (x++<=20);
}
```

x=1

o/p: 0 0 0 0 0

```
main()
{
  int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13,
  a14, a15;

  printf("Enter value 1:");
  scanf ("%d", &a1);

  printf("Enter value 2:");
  scanf("%d", &a2);

  printf ("Enter value 3:");
  scanf ("%d", &a3);

  printf ("Enter value 4:");
  scanf ("%d", &a4);

  printf ("Enter value 5:");
  scanf("%d", &a5);

  printf ("Enter value 6:");
  scanf ("%d", &a6);

  printf ("Enter value 7:");
  scanf ("%d", &a7);

  printf ("Enter value 8:");
  scanf ("%d", &a8);
```

```
printf("Enter value a9 :");
scanf("%d", &a9);

printf("Enter value 10 :");
scanf("%d", &a10);

printf("Enter value 11 :");
scanf("%d", &a11);

printf("Enter value 12 :");
scanf("%d", &a12);

printf("Enter value 13 :");
scanf("%d", &a13);

printf("Enter value 14 :");
scanf("%d", &a14);

printf("Enter value 15 :");
scanf("%d", &a15);

printf("%d", a1);
printf("%d", a2);
printf("%d", a3);
printf("%d", a4);
printf("%d", a5);
printf("%d", a6);
printf("%d", a7);
printf("%d", a8);
printf("%d", a9);
printf("%d", a10);
```

```
printf("%d",a[1]);
printf("%d",a[2]);
printf("%d",a[3]);
printf("%d",a[4]);
printf("%d",a[5]);
}
```

output: Enter value 1: 1
Enter value 2: 2
Enter value 3: 3
Enter value 4: 4
Enter value 5: 5
Enter value 6: 6
Enter value 7: 7
Enter value 8: 8
Enter value 9: 9
Enter value 10: 10
Enter value 11: 11
Enter value 12: 12
Enter value 13: 13
Enter value 14: 14
Enter value 15: 15

1
2
3
4
5
6
7
8

9
10
11
12
13
14
15

* Array: collection of homogeneous elements

```
main()
{
   int a[10];
   int sum=0, i;

   for(i=0; i<10; i++)
   {
      printf("\nEnter value %d:", i+1);
      scanf("%d", &a[i]);
   }

   for(i=0; i<10; i++)
   {
      printf("value %d : %d\n", i+1, a[i]);
      sum+=a[i];
   }
   printf("\nSum : %d", sum);
}
```

```
main()
{
  int a[10], i, sum=0;
  for(i=0; i<10; i++)
  {
    printf("\n Enter value %d :", i+1);
    scanf("%d", &a[i]);
    printf("value %d : %d\n", i+1, a[i]);
    sum += a[i];
  }
  printf("\nsum : %d", sum);
}
```

*
```
main()
{
  int b[5]; //array declaration
  int a[5] = {17, 29, 44, 56, 1}; //array initialization
```
Initializers.
```
  /*
   a[0] = 17;
   a[1] = 29;
   a[2] = 44;
   a[3] = 56;
   a[4] = 1;
  */
}
```

*
```
main()
{
  int a[0]; //ERROR
}
```
O/p: ERROR ⇒ Array must have at least one element

```
* main()
{
    int a[-33]; //ERROR : Array must have at
                          least one element
}

main()
{
    int a[]; //ERROR: size of 'a' is unknown or
}

* main()
{
    int a[] = {11, 22, 33, 44, 55, 66, 77}
    int i;

    for(i=0; i<7; i++)
        printf("a[%d] = %d \n", i, a[i]);
    printf("\n %d bytes", sizeof(a));
}
```

O/P: a[0] = 11
a[1] = 22
a[2] = 33
a[3] = 44
a[4] = 55
a[5] = 66
a[6] = 77

~~Size~~ 14 bytes.

maxi. size of array depends on near pointer
near pointer requires 2 bytes.

Page No.
Date

* main()
{
int a[10] = {11,22,33,44,55,66,77};
int i;
for (i=0; i<10; i++)
printf("a[%d]=%d\n", i, a[i]);
printf("%d bytes", sizeof(a));
}

zero

O/P: a[0] = 11
a[1] = 22
a[2] = 33
a[3] = 44
a[4] = 55
a[5] = 66
a[6] = 77
a[7] = 0
a[8] = 0
a[9] = 0
~~a[10]~~
20 bytes.

* main()
{
int a[5] = {11,22,33,44,55,66,77}; //ERROR:
}

ERROR: Too many initializers

```
main()
{
   int a[]={0,22,0,33,44,0,0,66};
   int i, count=0;

   for(i=0; i<8; i++)
   {
      printf("a[%d]=%d\n", i, a[i]);
       if(a[i]==0)
          count++;
   }
   printf("Array contains : %d zeros", count);
}
```

o/p:
```
a[0]=0
a[1]=22
a[2]=0
a[3]=33
a[4]=44
a[5]=0
a[6]=0
a[7]=66

Array contains : 4 zeros
```

*
```
main()
{
   int a[8];
   int i, cnt=0;

printf("\nEnter 8 elements:");
```

```
for(i=0; i<8; i++)
    scanf("%d", &a[i]); // input

printf("\nArray contains\n");
for(i=0; i<8; i++) // displaying an array
    printf("a[%d]=%d\n", i, a[i]);

for(i=0; i<8; i++) // calculating the total zeros
  if(a[i]==0)
      cnt++;

printf("\nArray contains : %d zeros", cnt);

}
```

o/p: Enter 8 elements : 0
22
0
33
44
0
0
66

a[0]=0
a[1]=22
a[2]=0
a[3]=33
a[4]=044
a[5]=0
a[6]=0
a[7]=66

Array contains : 4 zeros

* main()
{
int a[8] = {0,22,0,33,44,0,0,66};
int i, cnt=0;

for(i=0; i<8; i++)
{ printf("a[%d]=%d", i, a[i]);
if (a[i] & 1)
cnt++;
}
printf("Array contains : %d odd Nos.", cnt);
}

o/p: a[0] = 0
a[1] = 22
a[2] = 0
a[3] = 33
a[4] = 44
a[5] = 0
a[6] = 0
a[7] = 66

Array contains : 2 odd Nos.

* main()
{
int a[8] = {0,22,0,33,44,0,0,66};
int i, cnt=0, num=0;
int no=0;
for(i=0; i<8; i++)
{
printf("a[%d]=%d", i, a[i]);

```
        num=a[i];
        while(num)
        {
            no=num%10;
            if(no&1)
                cnt++;
            num/=10;
        }
    }
    printf("\nArray contains : %d odd digits", cnt);
}
```

* program for odd Nos. in Array:

```
main()
{
    int a[10];
    int i;
    printf("\nEnter 10 Nos.");
    for(i=0; i<10; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("\nArray Before Modification");
    for(i=0; i<10; i++)
    {
        printf("a[%d]=%d\n", i, a[i];
    }

for(i=0; i<10; i++)
{
    if(a[i]&1)
        a[i]=0;
}
```

```
printf("\nArray after Modification");
for(i=0; i<10; i++)
    printf("a[%d]=%d\n", i, a[i]);
}
```

O/P: Enter 10 Nos.

123
456
79
32
10
565
932
131
7
122

Array before Modification

a[0] = 123
a[1] = 456
a[2] = 79
a[3] = 32
a[4] = 10
a[5] = 565
a[6] = 932
a[7] = 131
a[8] = 7
a[9] = 122

Array
a[0]=
a[1]=
a[2]=
a[3]=
a[4]=
a[5]=
a[6]=
a[7]
a[8]
a[9]

* Linear

main()
{
int
int

fo
{
}

}

p

Array After modification
a[0] = 0
a[1] = 456
a[2] = 0
a[3] = 32
a[4] = 10
a[5] = 0
a[6] = 932
a[7] = 0
a[8] = 0
a[9] = 122

* Linear Search

```
main()
{
    int a[5] = {12, 90, 72, 30, 5};
    int i, num, flag = 1;

    for (i=0; i<5; i++)
    {
        printf("a[%d] = %d\n", i, a[i]);
    }

printf("\nEnter element For Search:");
scanf("%d", &num);

for (i=0; i<5; i++)
{
    if(a[i] == num)
    {
        flag == 1
        //printf("\n Entered element found");
        break;
    }
```

```
}else
if (flag==1)
    printf("\nEntered element found");
else
    printf("\nElement Not found");
}
```

⑧ Linear search without using any other variable.

```
main()
{
    int a[10];
    int i, no;

    printf("\nEnter 10 Elements:");
    for(i=0; i<10; i++)
    {
        scanf("%d", &no);
    }

    for(i=0; i<10; i++)
    {
        if(a[i]==num)
        break;
```

```
main()
{
    int a[10];
    int i, no, flag;

    printf("\n Enter 10 Elements :");
    for(i=0; i<10 ; i++)
        scanF("%d", &a[i]);

    printf("\n Enter Element to be deleted :");
    scanf("%d", &no);

    for(i=0; i<10; i++)
    {
        if(a[i]==no)
        {
            //flag=1;
            break;
        }
    }
    if(i!=10)
    {
        for(; i<9 ; i++)
        {
            a[i]=a[i+1];
        }
        printf("Entered element is deleted");
        for(i=0; i<9; i++)
        {
            printf("a[%d]=%d\n", i, a[i]);
        }
    }
    else
    {
        printf("Element not found for deletion");
        for(i=0; i<10; i++)
            printf("a[%d]=%d\n", i, a[i]);
    }
}
```

11	22	33	44	55	66	77	88	99	100

0	1	2	3	4	5	6	7	8	9
11	22	33	44	66	66	77	88	99	100

```
main()
{
    int a[10];
    int i, no, n, j

    printf("\n Enter No. of elements :");
     scanf("%d", &n);
```

Page No.
Date

Program for total no. of occurances of an element:

```
main()
{
    int a[10];
    int i, j, cnt=0, n;

    printf("\n Enter NO. of elements :");
    scanf("%d", &n);
                Entered
    printf("\n Elements in Array ~~are~~ :");
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("\n Elements in Array are :");
    for(i=0; i<n; i++)
    {
        printf(" a[%d] = %d \n", i, a[i]);
    }
```

11 22 33 11 44

```
For(i=0; i<0; i++)
{
    If(a[i] == a[i+1])
    {
        cnt++;
    }
    printf("Count of a[%d] = %d is : %d \n", i, cnt);
}
}
```

* Two Dimensional Arrays

```
main()
{
   int a[3][4]={
                {1,2,3,4},
                {5,6,7,8},
                {9,10,11,12}
               };
   int i,j;
   for(i=0;i<3;i++)
   {
      for printf("Row-%d\t\t", i+1);
         for(j=0; j<4; j++)
            printf("%d\t",a[i][j]);
         printf("\n");
   }
}
```

o/p:

Row-1	1	2	3	4
Row-2	5	6	7	8
Row-3	9	10	11	12

```
* main()
{
   int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12,13}; //ERRO
   int i,j;
   for(i=0; i<3; i++)
   {
      printf("Row-%d\t\t", i+1);
      for(j=0; j<4 ;j++)
         printf("%d\t",a[i][j]);
      printf("\n");
   }
```

ERROR: Too many initializers.

```
* main()
{
    int a[4][5];
    printf("%d bytes", sizeof(a));
}
```

O/p: 40 bytes.

```
* main()
{
    int test[][2] = {1,2,3,4,5,6};
    printf("%d bytes", sizeof(test));
}
```

O/p: 12 bytes.

```
* main()
{
    int test[][4] = {1,2,3,4,5,6,7,8,9};
    int i,j;
    for(i=0; i<3; i++)
    {
        for(j=0; j<4; j++)
            printf("%d\t", test[i][j]);
        printf("\n");
    }
    printf("%d bytes", sizeof(test));
}
```

//ERROR

O/p:

1	2	3	4
5	6	7	8
9	0	0	0

bytes.

ERROR: Too many initializers.

```
* main()
{
    int a[4][5];
    printf("%d bytes", sizeof(a));
}
```

O/p: 40 bytes.

```
* main()
{
    int test[][2] = {1,2,3,4,5,6};
    printf("%d bytes", sizeof(test));
}
```

O/p: 12 bytes.

```
* main()
{
    int test[][4] = {1,2,3,4,5,6,7,8,9};
    int i,j;
    for(i=0; i<3; i++)
    {
        for(j=0; j<4; j++)
            printf("%d\t", test[i][j]);
        printf("\n");
    }
    printf("%d bytes", sizeof(test));
}
```

O/p:

```
1   2   3   4
5   6   7   8
9   0   0   0
24 bytes.
```

```
main()
{
  int a[10][10];
  int i, j, row, col;

  printf("Enter Rows & columns in an array :");
  scanf("%d %d", &row, &col);

  printf("\nEnter %d elements :", row * col);

  for(i=0; i<row; i++)
    for(j=0; j<col; j++)
      scanf("%d", &a[i][j]);

  printf("\nYou Entered\n\n");
  for(i=0; i<row; i++)
  {
    printf("Row: %d\t\t", i+1);
    for(j=0; j<col; j++)
      printf("%d\t", a[i][j]);
    printf("\n");
  }
}
```

O/p: Enter Rows & columns in an array : 2
3

Enter 6 elements : 3
22
1
2
4
5

Row: 1 3 22 1
Row: 2 2 4 5

Que. For following O/P write a program:
Enter no. of rows & columns: 3
4

Enter 12 elements: 1
2
3
4
5
6
7
8
9
10
11
12

1 2 3 4 : 10
5 6 7 8 : 26
9 10 11 12 : 42

```
main()
{
    int a[10][10];
    int row, col, i, j, sum=0;

    printf("\nEnter no. of rows & columns:");
    scanf("%d %d", &row, &col);
```

```
printf("\nEnter %d elements : ", row * col);

for(i=0; i<row; i++)
    for(j=0; j<col; j++)
        scanf("%d", &a[i][j]);

for(i=0; i<row; i++)
{
    sum=0;
    for(j=0; j<col; j++)
    {
        sum = sum + a[i][j];
        printf("%d :\t", a[i][j]);
    }
    printf("%d", sum);
    printf("\n");
}
}
```

j=0 j=1 j=2 j=3
i=1 (1) 2 3 4

* Matrix Add^n & subtraction:

→

```
main()
{
    int a[10][10], b[10][10], c[10][10], d[10][10];
    int i, j, row, col, rows, cols,

    printf("Enter NO. of rows & columns for matrix : ");
    scanf("%d %d", &row, &col);

    printf("\nEnter %d elements : " row*col);
```

```
for(i=0; i<row; i++)
{
  for(j=0; j<col; j++)
    scanf("%d", &a[i][j]);
}
for(i=0; i<row; i++)
{
  for(j=0; j<col; j++)
    printf("%d\t", a[i][j]);
  printf("\n");
}

printf("Enter NO. of rows & colours for matrix b:");
scanf("%d %d", &row2, &col2);

printf("\nEnter %d elements:", row2*col2);
for(i=0; i<row2; i++)
  for(j=0; j<col2; j++)
    scanf("%d", &b[i][j]);

for(i=0; i<row2; i++)
{
  for(j=0; j<col2; j++)
    printf("%d\t", b[i][j]);
  printf("\n");
}

printf("\n Addition of two matrices is:");
for(i=0; i<row; i++)
{
  for(j=0; j<col; j++)
  {
    c[i][j]=a[i][j]+b[i][j];
    printf("%d\t", c[i][j]); //printf("%4d", c[i][j]);
  }
  printf("\n");
}
```

2=row
3=col
i=0
j=0

```
printf("\n Subtraction of two matrices ");
for(i=0; i<row; i++)
{
   For(j=0; j<col; j++)
   {
      d[i][j] = a[i][j] - b[i][j];
      printf("%d\t", d[i][j]); //printf("%4d", d[i][j]);
   }
   printf("\n");
}
}
```

* Matrix Multiplication:

```
main()
{
   int a[10][10], b[10][10], mul[10][10];
   int row1, col1, row2, col2, i, j, k

   printf("\nEnter no. of rows & coloumns in
                matrix 1:");
   scanf("%d %d", &row1, &col1);

   printf("\nEnter no. of rows & coloumns in
                matrix 2:");
   scanf("%d %d", &row2, &col2);

   if(col1 != row2)
   {
      printf("\n Multiplication is not possible");
   }
```

```
else
{
    printf("\nEnter %d elements of first matrix:",
                                    row1*col1);
    for(i=0; i<row1; i++)
        for(j=0; j<col1; j++)
            scanf("%d", &a[i][j]);

    ~~for(j=0; i<~~
    printf("\nEnter %d elements of 2nd matrix:",
                                    row2*col2);
    for(i=0; i<row2; i++)
        for(j=0; j<col2; j++)
            scanf("%d", &b[i][j]);

    for(i=0; i<row1; i++)
    {
        for(j=0; j<col2; j++)
        {
            for(k=0; k<col1 /*row2*/; k++)
            {
                mul[i][j]+= a[i][k] * b[k][j];
            }
        }
    }
    printf("\n Multiplication Matrix is:");
    for(i=0; i<row1; i++)
    {   for(j=0; j<col2; j++)
        {
            printf("%d\t", mul[i][j];
        }
        printf("\n");
    }
}
```

}

* program for Identity matrix

```
main()
{
    int a[i][j], i, j, flag=0, size;

    printf("\nEnter size of Identity matrix :");
    scanf("%d", &size);

    printf("\nEnter %d elements :" size*size);

    for(i=0; i<size; i++)
        for (j=0; j<size; j++)
            scanf("%d", a[i][j]);

    for(i=0; i<size; i++)
    {
        for(j=0; j<size; j++)
        {
            if (i==j)  //diagonal elements
            {
                if (a[i][j]!=1)
                {   flag=1;
                    //printf("Matrix is not Identity matrix");
                }
            }
            else
            {
                if (a[i][j]!=0)
                {   flag=1
                    //printf("\n Matrix is Identity");
                }
    }
```

```
    }
    if(flag==1)
    {
      printf("\nNot an Identity matrix");
    }
    else
      printf("Identity Matrix);
  }
}
```

* Spiral Matrix:

```
int iter,j,k=1,n;
int a[21][21];

printf("\nEnter size of Spiral matrix:");
scanf("%d",&n);

for(iter=0;iter<=n/2;iter++)
{
  for(j=iter;j<n-iter;j++)
  {
    a[iter][j]=k++;
  }
  for(j=iter+1;j<n-iter;j++)
  {
    a[j][n-1-iter]=k++;
  }
  for(j=n-2-iter;j<=>=iter;j--)
  {
    a[n-1-iter][j]=k++;
  }
}
```

```
for (j= n-2-iter ; j >iter ; j--) ; j--)
{
    a[j][n-2-iter] = k++ ;
}
}
printf("\n\n\nSpiral Matrix :\n\n");
for (j=0 ; j<n ; j++)
{   for(k=0;k<n;k++)
    printf("%4d \t", a[j][k]);
}
printf("\n");
}
```

O/P:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

Magic Square:

1. put 1st element in 1st row middle position
2. 1 step right & 1 step up
3. If element already placed then go back to previous element & place new element below previous element.

```
main()
{
```

0	8	1	6
1	3	5	7
2	4	9	2

n=3

a[0][n/2] = k;

For[

```
main()
{
   int i, j, n,
   a[21][21]={0};

   ~~a[i][j]~~
   printf("\nEnter size of Magic Square :");
   scanf("%d", &n);

   ~~if~~ ~~n/2==~~
```

```
main()
{
  int magic[21][21]={0};
  int cr, cc, pr, pc, k=1, n, sum=0;

  printf("\nEnter an odd size:");
  scanf("%d", &n);
  if(!(n & 1))
  {
    printf("\nAre you mad??(Yes/NO):");
    exit(1);
  }

  magic[0][n/2]=k++; //step 1
  cr=0;
  cc=n/2;

  while(k<=n*n)
  {
    pr=cr;
    pc=cc;
```

```
        cc++;  //step 2
        if (cc==n)  //rule1
            cc=0;

        cr--;  //step3
        if(cr==-1)  //rule 2
            cr=n-1;

        if (!magic[cr][cc])
        {
            magic[cr][cc]=k++;
        }
        else  //if element is already present
        {
            cr=pr+1;  //rule 3
            cc=pc;
            magic [cr][cc]=k++;
        }
    }
    printf("\nMagic Square of size %d*%d\n\n",n);
    for(cr=0; cr<n ;cr++)
    {
        sum+=magic[0][cr];
        for(cc=0; cc<n; cc++)
            printf ("%4d", magic[cr][cc]);
        printf("\n");
    }
    printf("\n\nSum of each row/column/Diagonal:%d",sum);
}
```

'0' ⇒ 48

String:

```
main()
{
  char a[] = {'a', 'b', 'c', 'd', 'e', '\0'};
  printf("String : %s", a);
}
```

o/p: String : abcde

```
main()
{
  char a[10] = {97, 98, 99, 100, 0, 101, 102};
  int i;                          ↑ ascii value of '\0'

  for(i=0; i<7; i++)
      putchar(a[i]);
}
```

o/p: abcd ef

```
main()
{
  char a[10] = {97, 98, 99, 100, 0, 101, 102,};
  //puts(a);
  printf("%s", a);
}
```

o/p: abcd

```
main()
{
  char str[] = "This is a test line";
  printf("String : %s", str);
  printf("\n\n%d bytes", sizeof(str));
}
```

o/p: This String = This is a test line.

20 bytes

* main()
{
char str[30] = "This is a test line\0";
printf("\nEnter a string : ");
gets(str); // this
str[4] = 'a';
printf("You Entered : %s", str);
}

o/p:

Enter a string : this
You Entered : thisais a test line.

* To find string length:

```
main()
{
  int str[30];
  int i;

  printf("\nEnter a String :");
  gets(str);
  i=0;
  while (str[i++]);

  printf("\nLength : %d", --i);
}
```

* program to copy string:

```
main()
{
    int str[30], str1[30];
    int i;

    printf("\n Enter a string:");
    gets(str);

    i=0;

    while(str[i++])
    {
        str1[i]=str[i]; i++;
    }   printf("\n copied strin
    str1[i]='\0';
    printf("\n copied string is: %s", str1[i]);

}
```

* program to compair length of strings:

```
main()
{
    int str[30], str1[30];
    int i, j, len1, len2;

    printf("\n Enter a string1:");
    gets(str);

    i=0;
    while(str[i++]);
    len1=i--;
```

```
printf("\n Enter a string 2:");
gets (str1);

j=0;
while (str[j++]);
 or len2=j--j;
if (i==j) (len2==len1)
    printf ("\n Strings are of Equal length");
else
    printf ("\n Strings are of different length");

}
```

* program for concatenation of strings

```
#include <stdio.h>
int main()
{
  char st
  int str1[30], str2[30];
  int i, j, len=0;

printf("\n Enter first string:");
gets(str1);

/*  i=0;
while (str1[i++]);
len = --i;

j=0;
while (str2[j])
```

OR

```
i=0;
while (str1[i])//move
    i++;
j=0;
while(str1[i]=str2[j]
{
    i++;
```

```
{
    str1[len] = str2[j];
    j++;
    len++;
}
*/
printf("\n concatenated string is : %s", str1);
}
```

* program for compairing strings:

```
main()
{
    int str1[30], str2[30];
    int i, j, flag=0;

    printf("\nEnter first string :");
    gets(str1);

    printf("\nEnter second string :");
    gets(str2);
    j=0;
    i=0;
    while( while(str1[i] || str2[j])      abc
    {                                     abcd
    if(str1[i] == str2[j])
    {   flag=1;
        i++;
        j++;
```

```
        else
        {flag=0;
         break;
        }
    }
    if(flag==1)
        printf("\n Strings are Equal");
    else
        printf("\n strings are not equal");
}
```

* program to reverse string:

```
main()
{
    int str[30], rev[30];
    int i, j;
    printf("\n Enter a string:"); gets(str);
    i=0;
    while(str[i])
        i++;

    j=0;
    for(i=i-1; i>=0; i--)
    {
        rev[j]=str[i];
        j++;
    }
    rev[j]='\0';
    printf("\n Rerersed string is: %s", rev);
}
```

program For Palindrome:
level.

```
main()
{
   int str1[30], str2[30];
   int i, j, flag = 0;

   printf("\n\nEnter a string:");
   gets(str1);

   i = 0;
   while (str[i])
        i++;

   j = 0;
   for(j; i = i-1 ; i >= 0; i--)
   {
      str2[j] = str1[i];
      j++;
   }
   str2[j] = '\0';

   i = 0; j = 0;
   while (str1[i] || str2[j])
   } {
   if (str1[i] == str2[j])
    {
      flag = 1;
      i++;
      j++;
   }
   else
   {
```

```
        flag=0;
        break;
      }
    }
    if(flag==1)
        printf("\n Strings are palindrome");
    else
        printf("\n Strings are not palindrome");

}

* main()          //program for compaining strings
{
  char str1[30], str2[30];
  int i=0;

  printf("Enter first string:");
  gets(str1);

  printf("\n Enter second string:");
  gets(str2);

  while (str1[i] && str2[i] && !(str1[i]-str2[i]) &&+i);
  !(str1[i]-str2[i]) ? printf("same") : printf
                                ("\n Different");
}
```

* program for Reverse string

```
main()
{
    char str[30], temp;
    int i, j, temp;

    printf("\nEnter a string :");
    gets(str);

    i=0;
    while (str[i]) i(str[i++);
    j=i-1;

    i=0;
    while(i<j)
    {
        temp=str[i];
        str[i]=str[j-1];
        str[j-1]=temp;
    }   i++; j--;
    printf("\nReverse string is :%s", str);
}
```

* program for Palindrome:

```
main()
{
    int str[30], str1[30];
    int i, j, temp;

    printf("\nEnter a string :");
    gets(str);
```

```
i=0;
while(str[i++]);
j=i-1;

i=0;
while (i<j)
{
    temp=str[i];
    str[i]=str[j-1];
    str[j-1]=temp;
    i++; j--;
}

i=0;                    // for copy
while(str[i])
{
    str1[i]=str[i];
    i++;
}
str1[i]='\0';

// while ( str[i] && str1[i] && !(str[i]-str1[i]
                    && ++i);

!(str[i]-str1[i]) ? printf ("Palindrome");
                  printf("Not palindrom;
}
```

Scanned by CamScanner

Reverse String:

```
main()
{
    char str[30], temp;
    int i, j;

    printf("\n Enter a string:");
    gets(str);

    j=0;
    while(str[j])
        j++;
    j--;

    i=0;
    while(i<j)
    {
        temp=str[i];
        str[i]=str[j];
        str[j]=temp;
        i++;
        j--;
    }
    printf("\n Reverse string is :%s", str);
}
```

* palindrome string:

```
main()
{
    char str[30];
    int i,j;

    printf("\n Enter a string");
    gets(str);

    j=0;
    while(str[j]) //move 'j' on the last character
        j++;
    j--;

    i=0;
    while(i<j)
    {
        if(str[i]!=str[j])
            break;
        i++;j--;
    }

    if(i>=j) //normal termination
        printf("\n Palindrome");
    else
        printf("\n Not a palindrome");
}
```

Interpolation

To find substring:

```
main()
{
    char str[30], sub[30], str1[30];
    int i, j;

    printf("\nEnter a string :");
    gets(str);

    printf("\nEnter a substring :");
    gets(sub);

    i=0, j=0;
    while(str[i] && sub[j])
    {
        if(str[i]==sub[j])
        { str1[i]=str[i];
          i++; j++;
        }
        else
        {
          j=0; i++;
        }
    } str1[i]='\0';
    if(str1[i]==sub[j])
        printf("\n Substring is found");
    else
        printf("\n Substring is not found");
}
```

abcdefghijkl

fgh

abcde

Scanned by CamScanner

Anagrams = First strings characters are taken to form second string. e.g. creative & reactive

```
main()
{
   char str1[30], str2[30];
   int *visited, num;

   printf("\nEnter first string :");
   gets(str1);

   printf("\nEnter second string :");
   gets(str2);

   num=0;
   while (str2[num])
       num++;

   visited= malloc (size.of(int)*num);

   for(i=0 ; i<num ; i++)
       visited[i]=0;

   for(i=0; ~~i<num~~ str1[i] ; i++)
   {
      for(j=0; ~~j<num~~ str2[j] ; j++)
      {
       if (str1[i]==str2[j] && !visited[j])
       {
          visited[j]=1;
          break;
       }
   }
}
```

```
        if (str2[j] == '\0')
        {
            break;
        }
    }

    if(str1[i] == '\0')
    {
        for(i=0; i<num; i++)
            if (visited[i]==0)          // if(!visited[i])
            {
                printf("\nNot an Anagrams");
                exit(1);
            }
        printf("\nAnagrams");
    }
    else
        printf("\nNot an Anagrams");
}
```

Page No.
Date

Functions:

```
void fact (int n) // Function definition
{
  long f=1L;
  while (n)
      f *= n--;
  printf("%d\n", f);
}
main()
{
  int i;
  for(i=1; i<=10; i)
  {
     printf("%d! :", i);
     fact(i); // Function call
  }
}
```

O/p: 1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800

* Arrange array elements in ascending order:

```
int findmin (int x[], int n)
{
  int i, min=9999, minIndex;
  for(i=0; i<n; i++)
     if(min > x[i])
     {
       min=x[i];
       minIndex = i;
     }
  return minIndex;
}

main()
{
  int a[] = {13, 44, 66, 2, 77, 8, 9, 79, 12, 11};
  int i, z;
  for(i=0; i<10; i++)
  {
     z=findmin(a,10); // passing an array to a fu^n
     printf("%d", a[z]);
     a[z]=9999; // set found element to infinity
  }
```

O/p: 2 8 9 11 12 13 44 66 77 79

*

```
void showMessage(char str[], int n)
{
  int i;
  for(i=0; i<n; i++)
     printf("%s\n", str);
}
```

```
main()
{
showMessage("This", 4);
showMessage("Hi", 3);
showMessage("wait", 5);
showMessage("cycle", 2);
}
```

O/P:
This
This
This
This
Hi
Hi
Hi
wait
wait
wait
wait
wait
cycle
cycle.

* Syntax of fun:

```
return-type function-name(parameter-list)
{
}
```

```
int max(int x, int y)   // function body.
{
    if (x > y)
        return x;
    else
        return y;
}
main()
{
  int z;
  z = max(10,11);  // function call
  printf("Max = %d", z);
}
```

O/p: Max = 11

```
void show();        // function declaration
main()
{
    show();    // Function call
    show();
    show();
}
void show()
{
   printf("Inside show()");
}
```

O/p: Inside show()
Inside show()
Inside show()

* float max(float, float); //declaration
main()
{
printf("%f", max(123.456f, 123.457f));
}
float max(float x, float y)
{
if (x>y)
return x;
else
return y;
}

O/p: 123.457001

* float max(float, float);
main()
{
printf("%f", max(123.456f, 123.457f));
}
float max(float x, float y)
{
return x>y ? x : y;
}

O/p: 123.457001

formal arguments

```
int min(int x, int y) // formal arguments
{
   return x<y ? x : y;
}
main()
{
  int x;
  printf("\nEnter a number:");
  scanf("%d", &x);
  printf("Min: %d", min(x,11) /*Actual argu*/); //OK
  printf("Min: %d", min(x,11,22)); //Extra
                                    parameter
  printf("Min: %d", min(x)); // Too few parameters
}

float test4(float x)
{
   printf("inside test4()\n");
   return x+1.1f;
}
float test2(float x)
{
   printf("inside test2()\n");
   return test4(x)+1.1f;
   printf("inside test2() again\n");
}
float test3(float x)
{
  printf("inside test3()\n");
  return test2(x)+1.1f;
  printf("inside test3() again\n");
}
```

```
float test1(float x)
{
    printf("\n inside test1 ");
    return test3(x) + 1.1f;
    printf("\n inside test1() again");
}
main()
{
    printf("Inside main()\n");
    printf("%.2f\n", test1(1.1f));
    printf("inside main() again \n");
}
```

O/p: Inside main()
inside test1()
inside test3()
inside test2()
inside test4()
5.50
inside main() again.

Passing An Array to the function.

Page No.
Date

```
int findMax (int x[], int n)
{
  int i, max;
  max = x[0];
  for(i=1; i<n; i++)
      if (x[i] > max)
          max = x[i];
  return max;
}
main()
{
  int a[] = {33, 22, 11, 44, 99, 77, 88, 101, 55};
  printf("Max = %d", findMax(a, 9));
}
```

O/p: Max = 101

* Linear Search & return index of searched element

```
int ~~find~~ LSearch(int x[], int n, int no)
{
  int i;
  for(i=0; i<n; i++)
  {
      if (x[i] == no)
          return no;
      else
          printf("\nElement not found");
```

```
main()
{
    int a[] = {
```

*
```
int input(int a[], int maxEle)
{
    int i, n;
    printf("\nHow many elements : (MAX: "
    scanf("%d", &n);

    printf("\nEnter %d elements : ", n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    return n;    // no. of elements
}
int search (int a[], int n, int k)
{
    int i;
    for(i=0; i<n; i++)
        if(a[i]==k)
            return i;
    return -1; // if not found
```

```
}
void del (int a[], int n, int k, int index)
{
  int i;
  for(i=index; i<n-1; i++)
     a[i] = a[i+1];
}
void show (int a[], int n)
{
   int i;
   if (n==0)
   {
      printf("\n\nNothing to display ...");
      return; // return control to the calling fun.
   }
   printf("\n\nArray\n");
  for(i=0; i<n; i++)
      printf("a[%d] = %d\n", i, a[i]);
%d], maxEle);
   }

main()
{
   int a[25], n, key, index, flag=1;
   n = input(a, 25);

   printf("\nEnter an element to be deleted :");
   scanf("%d", &key);

   do
   {
      index = lsearch(a, n, key);
```

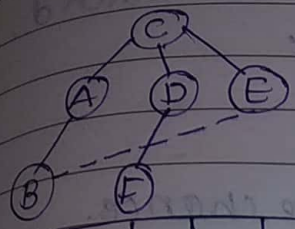
```
if(index==-1)
{
    if(flag==1) // if at all not found
        printf("\n Not found..\n");
    break;
}
else
    del(a,n--,key,index);
flag++;
}while(1);

show(a,n);
}
```

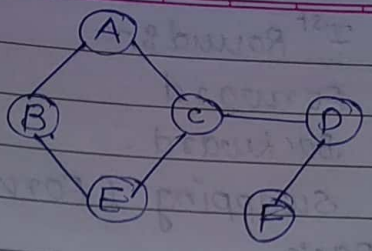
```
* Void add(int,int);
  void test();
  main()
  {
     int x=44, *y=55;
     add(x,y);
     test();
  }
  void test()
  {
    add(11,22);
  }
  void add(int a,int b)
  {
    printf("\n sum: %d", a+b);
  }
```

o/p: sum=99

sum=33

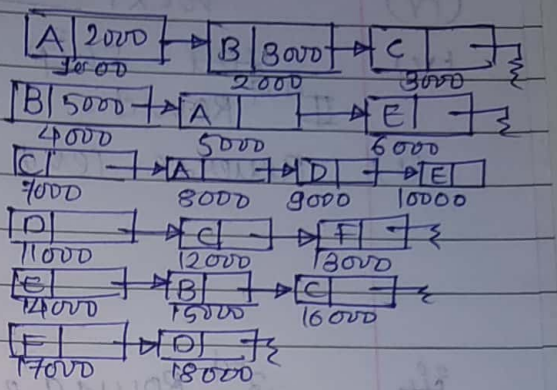
BFS = CADEBF





queue = | C | A | D | E | B | F |

1000	
4000	
7000	
11000	
14000	
17000	



```
* void test();
  main()
  { int x=44, y=35;
    void add(int x, int y); // 'add' is declared locally
    add(x,y);                  (just for main())
    test();
  }
  void test()
  { add(11,22); // ERROR: Type mismatch in redeclaration
                                of array
  }
  void add(int a, int b)
  { printf("\n sum: %d", a+b);
  }
```

ERROR: Because calling fu.n i.e. test() is before add i.e. called fu.n