24/08/23

Primary Memory → Volatile

Secondary Memory → Non - Volatile

~~Comp~~

Number System

- Binary Number system (Base 2)
- Decimal Number system (Base 10)
- Hexadecimal Number System (Base 16)
- Octal Number system (Base 8)

Binary : 0, 1

Decimal : 0, 1, 2, ... 9

Octal : 0, 1, 2, ... 7

~~Hexd~~

Hexadecimal : 0, 1, 2, ... 9, A, B, ..., F

→ (A : 10, B : 11, ... F : 15)

Q) $(865)_8$ → Valid or not

Sol : Not Valid ∵ 8 → Not Valid

Q) $(a1B)_{16}$ → Valid or Not

Sol : Not Valid ∵ a → Should be Capital.

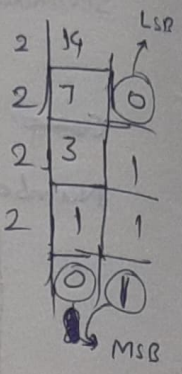
Q) $(012)_4$ → Valid or not

Sol : Yes, Valid [0, 1, 2, 3]

Conversion of Decimal to Binary Number system.

Q) Convert $(14)_{10}$ to Binary

$$(14)_{10} \equiv (1110)_2$$



Q) Convert $(57)_{10}$ to Binary

2	57	
2	28	1
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

$$(57)_{10} \equiv (111001)_2$$

$2^8\ 2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Practice:

1) $(512)_{10} \equiv (1000000000)_2$

$(257)_{10} \equiv (100000001)_2$

C:

1) Windows Search "Terminal"

command 2) gedit filename.c

3) Save and close editor

command 4) cc filename.c

command 5) ./a.out - Run

Starting of C :

(i)

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

[returntype function ( )]

(ii) For New Line, (Tutorial0.c)

```
printf ("Hello World\n");
printf ("How are you ?");
```

(iii) Variable, (Tutorial01.c)

```
int a;
a=5;
int b=10;
int c;
c=a+b;
printf ("The Sum is %d", c);
```

(iv) Taking data (Tutorial02.c)

```
int a,b,c;
printf("Enter the first number \n");
scanf("%d", &a)
printf("Enter the second number \n");
scanf("%d", &b)
c = a+b
printf("The sum is %d, c);
```

(v) Subtraction (Tutorial 03.c)

```
#include<stdio.h>
void main( )
{
    int a,b,c ;
    printf ("
    scanf
    printf
    scanf
    c = a - b
    printf
```

Q) Write a C programme to convert the temparature from Celsius to Farhenheit. (Tutorial.c)

Q) Write a C program to compute the average of 4 numbers (Tutorial1.c)

Q) Write a C program to compute the area of Rectangle (Tutorial2.c)

Circle → Tutorial31.c
Square → Tutorial3.c
Triangle → Tutorial32.c
Cylinder → Tutorial4.c
Sphere → Tutorial5.c
Hemisphere → Tutorial6.c

25/08/23

→ Converting Decimal to Octal

Ex. $(243)_{10} \equiv (363)_8$

8	243	
8	30	3
8	3	6
	0	3

Q) $(3402)_{10} \equiv (\ )_8$?

$(3402)_{10} \equiv (6512)_8$

8	3402	
8	425	2
8	53	1
8	6	5
	0	6

→ Converting Decimal to Hexadecimal :

Ex. $(243)_{10} \equiv (\ )_{16}$?

$(243)_{10} \equiv (F3)_{16}$

16	243	
16	15	3
	0	15

Ex. $(5062)_{10} \equiv (\ )_{16}$?

$(5062)_{10} \equiv (13C6)_{16}$

16	5062	
16	316	6
16	19	12
	1	3

→ Converting Binary to Decimal :

Ex. $(100111)_2 \equiv (\ )_{10}$?

	1	0	0	1	1	1
×	2^5	2^4	2^3	2^2	2^1	2^0

$32 + 4 + 2 + 1 = 39$

$(100111)_2 \equiv (39)_{10}$

Q) $(11111)_2 \equiv (\ )_{10}$

$16 + 8 + 4 + 2 + 1 = (31)_{10}$

2	39	
2	19	1
2	9	1
2	4	1
2	2	0
2	1	0
	0	1

2	31	
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

Q) $(10101010)_2 \equiv ?_{10}$

$128 + 32 + 8 + 2 = 170$

— Converting an Octal no to Decimal no :

Ex $(763)_8 \equiv (\ )_{10}$?

$(7 \times 8^2) + (6 \times 8) + (3 \times 1)$ $64 \times 7 = 448$

$= 448 + 48 + 3$

$= 499$

Q) $(169)_8 \equiv (\ )_{10}$

Not a Valid Octal No.

Q) $(1652)_8 \equiv (\ )_{10}$ 64×6

$= 512 + 384 + 40 + 2$

$= 938$

— Conversion of Hexadecimal no. to Decimal no. :

Ex. $(763A)_{16} \equiv (\ )_{10}$

$7 \times 16^3 + 6 \times 256 + 48 + 10$

$\Rightarrow 1594 + 7 \times 16^3$

$\Rightarrow 30266$

256
6
1536

$(10+6)^3$

1000
36
1800
1080
3916

27412
1594
29006

6 4
3916
7
27412

Ex. $(F12)_{16}$

$(15 \times 256) + 16 + 2$

$= (3858)_{10}$

256 (10+7)

2560
1280
3840

2 | 256
2 | 128
2 | 64
2 | 32
2 | 16
2 | 8
2 | 4
2 | 2
1

→ Converting Binary No to Octal No.

Ex. $(101101111)_2$

$256 + 64 + 32 + 8 + 4 + 2 + 1 = (367)_{10}$

$(367)_{10} \equiv (557)_8$

8	367	
8	45	7
8	5	5
	0	5

~~Ex~~.

Shortcut:

$(101\,101\,111)_2 \equiv (\ )_8$?

$(101)_2 \equiv (5)_{10}$
$(101)_2 \equiv (5)_{10}$
$(111)_2 \equiv (7)_{10}$
$\Rightarrow (557)_8$

(3 digit sets from right)

Q) $(101011)_2 \equiv (\ )_7$

$32 + 8 + 2 + 1 = 43$

7	43	
7	6	1
	0	6

$(101011)_2 \equiv (43)_{10} \equiv (61)_7$

Q) $(10101011)_2 \equiv (\ )_{16}$

$(1010)_2 = 10 \equiv A$
$(1011)_2 \equiv 11 \equiv B$

$(AB)_{16} \equiv (10101011)_2$

Q) $(1010.11)_2 = (\ )_{10}$?

1	0	1	0	.	1	1
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}

$(10.75)_{10}$

→ Operations:

(1) Arithmetic operator

(2) Relational Operator

(3) Logical Operators

&& → And

|| → Or

! → Not

(4) Assignment Operator

$a = b \Rightarrow a = b$

$a += b \Rightarrow a = a + b$

$a -= b \Rightarrow a = a - b$

$a *= b \Rightarrow a = a * b$

$a /= b \Rightarrow a = a / b$

$a \%= b \Rightarrow a = a \% b$

$a += b \times c \Rightarrow a = a + (b \times c)$

$a \times = b + c \Rightarrow a = a \times (b + c)$

(5) Increment and Decrement Operators

Post Increment : a++	Post decrement : a--
Pre Increment : ++a	Pre decrement : --a

Ex. $a = 5$, $y = a++$ ⟹ ① $y = a$ ② $a = a + 1$ ⇒ $y = 5$, $a = 6$

Ex. `a = 5;`

`printf("%d", a++);` ⟶ Result = 5

`printf("%d", a);` ⟶ Result = 6

Ex. $a = 5$, $y = ++a$ ⟹ ① $a = a + 1$ ② $y = a$

p(y) ⟶ Result = 6

p(a) ⟶ Result = 6

`printf("%d", ++a)` ⟶ Result = 6

(6) Conditional Operator :

Ex. $a = 5$, $b = 6$

`(a < b) ? Printf("Yes") : Printf("No");`

~~`Printf("No")`~~

Ex. Even / Odd

`#Include`

(7) Bitwise Operators (Only on Integers)

(a) Bitwise AND (&)
(b) Bitwise OR (|)
(c) Bitwise NOT
(d) Bitwise EXCLUSIVE OR (^)
(e) Left shift (<<)
(f) Right shift (>>)

(a)

A	B	A & B
1	1	1
1	0	0
0	1	0
0	0	0

a = 4
b = 5
Printf("%d", a&b)

```
  100
  101
  ---
  100
```

Result : 4

Ex. a = 16
b = 7
Printf("%d", a&b)

```
  10000
  00111
  -----
  00000
```

Output : 0

```
  010
  100
  ---
  110 => 6
```

(b)

A	B	A | B
1	1	1
1	0	1
0	1	1
0	0	0

Ex. a = 16
b = 64
Printf("%d", a|b)

```
  0010000
  1000000
  -------
  1010000
```

↳ Output = 80

(d)

A	B	A ^ B
1	1	0
1	0	1
0	1	1
0	0	0

$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$

~~Exclusive OR~~

[+ → Bitwise OR
. → Bitwise AND
⊕ → Bitwise EXCLUSIVE OR]

Ex. a = 4
b = 5
Printf("%d", a ^ b)

Output : 1

~~010~~ 100
~~100~~ 101
001

Ex. a = 10
b = 6
Printf("%d", a ^ b)

Output : 12

1010
0110
1100

2	10	
2	5	0
2	2	1
	1	0

(e) a << b

16 << 1

1 0 0 0 0
↙ ↙ ↙ ↙ ↙
_ _ _ _ _ 0

Output : 32

7 << 2

111 } → Output : 28
11100 }

★ Shortcut:

$a << b \Rightarrow a \times 2^b$

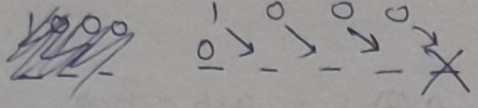
$7 << 2 \Rightarrow 7 \times 2^2 = 28$

??

(d) $a >> b$

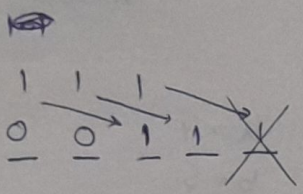
Ex. $8 >> 1$

Output: 4

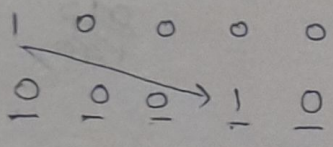


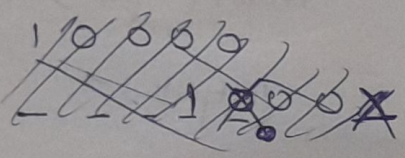
Ex. $7 >> 2$

Output: 3



Ex. $16 >> 3$





Output: 2

★ Shortcut:

$a >> b \Rightarrow a/2^b$

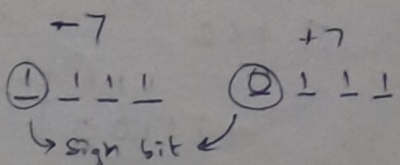
$16 >> 3 \Rightarrow 16/2^3 = 2$

[Quotient]

(e) Sign Magnitude Representation
Ones Complement Representation
Twos Complement Representation

Sign Bits:

positive → 0
negative → 1



Ex. -16

1 10000

+0 -0

0000 1000

Ex. -5

1 1 0 1

Negative numbers can never be represented using Sign Magnitude Representation.

31/8/23

Algorithm:

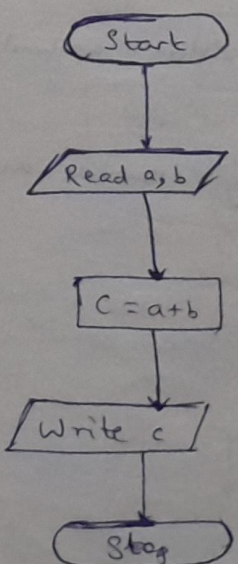
Input → Output

Definiteness ✓
Finiteness ✓
Effectiveness ✓

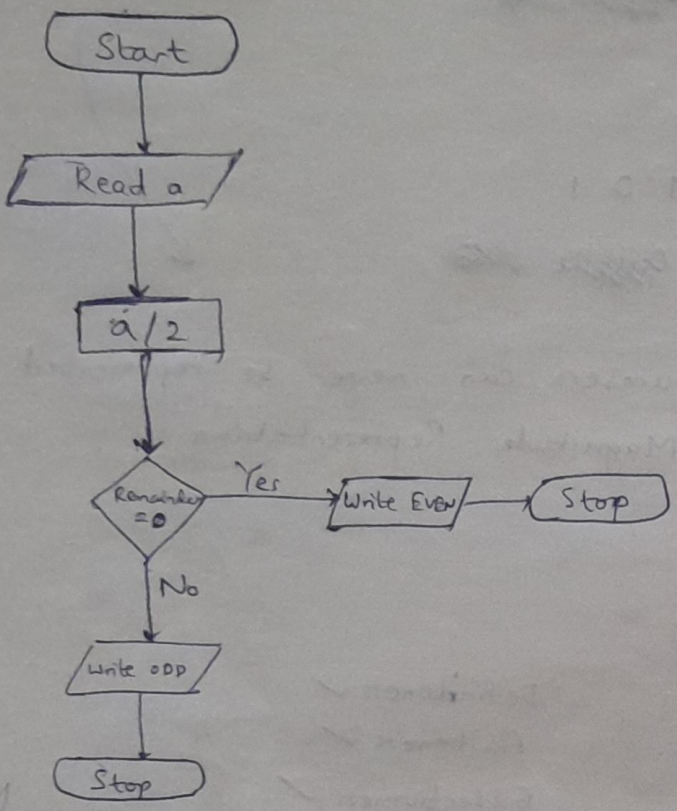
Algorithm → Start ... Stop

Sequence | Selection | Iteration

Ex. Draw a flowchart for finding sum of 2 numbers



Ex. Draw a flowchart to verify if a number is even or odd.



Q1) Swap 2 numbers using only Third variable

Q2) Swap 2 numbers without using Third Variable

Q3) Check whether a given number is even or odd using a ternary operator.

Q4) Find whether a number is divisible by 7 or not

Q5) Swap 2 numbers using Multiplication & Division without third Variable

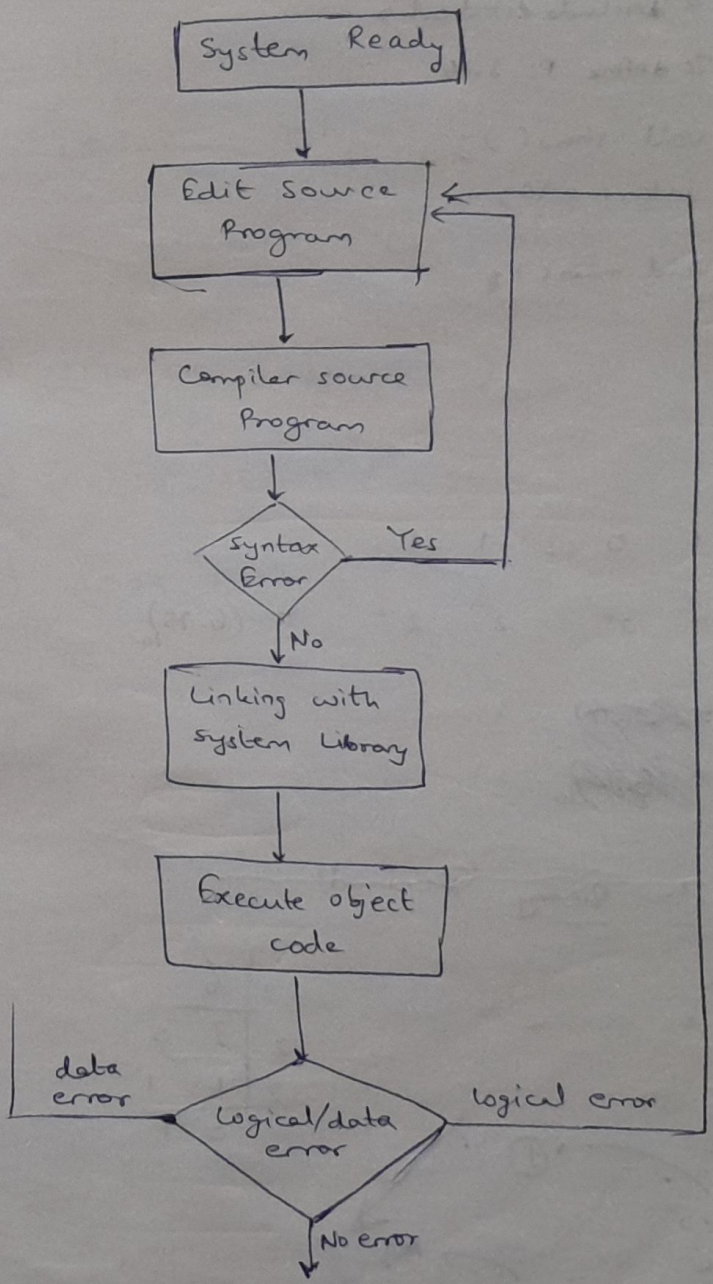
Q6) Find the sum of 4 digits of a 4 digit number and Also show reverse of the number.

1/9/23

Storing code data → Secondary Memory
Executing Program → Primary / Main Memory

Editing → Compiling → Linking → Executing
(gedit) (cc) (./a.out)

High Level Language —Compiler→ Low Level Language



→ Structure of C Program:

Documentation

Link Section

Definition section

Global Declaration Section

Main Function

Subprogram Section

Functions
- Pre Defined
- User Defined

Documentation : // sample (or /* sample */)

Link : #include<stdio.h>

Definition : #define Pi 3.14

Global Declaration : void show();
int a=10;

Main Function : Void main(){

Q) $(110.11)_2 \equiv (\ )_{10}$?

1 1 0 . 1 1

2^2 2^1 2^0 2^{-1} 2^{-2} $\equiv (6.75)_{10}$

$/10$

→ Converting Decimal to Binary (Decimal)

$(6.75)_{10} \equiv (?)_2$

$(6)_{10} \equiv (110)_2$

2	6
2	3 0
2	1 1
	0

$0.75 \times 2 \Rightarrow 1.5$ → (1)

$0.5 \times 2 \Rightarrow 1.0$ → (1)

$0.0 \times 2 = 0$

$(0.75)_{10} \equiv (0.11)_2$

1 ✓ 0
1 1 ↓

$(0.11)_2$ $(0.01)_2$

(Q) $(15.125)_{10} \equiv (\quad)_2$

$(15)_{10} \equiv (1111)_2$

2	15		
2	7	1	
2	3	1	
2	1	1	
		1	

$\boxed{(1111.001)_2}$

$0.125 \times 2 = 0.25$ → 0

$0.25 \times 2 = 0.5$ → 0

$0.5 \times 2 = 1.0$ → 1

$0.0 \times 2 = 0$

$(15)_{10} \equiv (1111)_2$

$(0.125)_{10} \equiv (0.001)_2$

$(15.125)_{10} \equiv (1111.001)_2$

$Q_1)$ $(111345.25)_8 \equiv (\quad)_{10}$

$Q_2)$ $(ABC12.35)_{16} \equiv (\quad)_{10}$

$Q_3)$ $(1025.125)_{10} \equiv (\quad)_8 \equiv (\quad)_{16}$

5/9/23 :

C Tokens :

1) Keywords
2) Identifiers
3) Constants
4) Strings
5) Special Symbols
6) Operators

(1) Keywords

auto	double	int	struct	break	else
case	enum	register	typedef	char	extern
const	short	float	unsigned	continue	for
default	goto	sizeof	volatile	do	if

long	switch
return	union
signed	void
static	while

Keywords → Always in Lowercase
Keywords → Can't be used as Variable name

(2) ~~Variable~~ Identifier :

(i) The ~~first~~ first character always should be an Alphabet or an Underscore.

(ii) Formed using only Alphabet, Number (or) Underscore.

(iii) Keyword cannot be used as an Identifier

(iv) Should not contain space character.

(3) Constants :

Fixed Value, The program can't change the value during its execution.

Syntax :

```
const int a = 10
a = a+1
    ↳ error
```

```
int a = 10
a = a+1
    ↳ Output : a = 11
```

(i) Must not have a decimal point. It can be either positive or negative.

(ii) No comma between Integers digits.

\b ⟶ Backspace
\f ⟶ form feed
\n ⟶ new line
\r ⟶ carriage return
\t ⟶ Horizontal tab
\v ⟶ vertical tab
\? ⟶ Question mark
\" ⟶ Double Quote
\' ⟶ Single Quote
\\ ⟶ Backslash
\a ⟶ Alert (or) Bell
\N ⟶ Octal Constant
\XN ⟶ Hexadecimal constant
\0 ⟶ Null Character

(4) Strings :

Strings are always in " ".

(5) Special Symbols :

{ } — Grouping zero or more statements into a block.

( ) — Specifying a precedence change in an expression.

[ ] → Declaring / Accessing an array element's value

; — Seperating one statement from another.

, → Seperating variables

(6) Variable :

Name of memory location

Rules:

(i) Variable can have alphabets, digits and underscore.

(ii) Can't start with a digit

(iii) Blank space is allowed within the variable name.

(iv) Keywords cannot be a variable.

(v) Case Sensitive.

One's Complement :

1 0 1 0

↓

0 1 0 1

Ex. One's Complement of −5

Step - I : Represent +5 firstly

0 1 0 1

+5:
0101
r0:0
10:1

Step - II : Find One's Complement

1 0 1 0

Q) −26 in One's Complement Representation

0 1 1 0 1 0

↓

1 0 0 1 0 1

2	26	
2	13	0
2	6	1
2	3	0
2	1	1
	0	1

Q) +0 , −0 in one's complement Representation

0 0 0 0

↓

1 1 1 1

0 0 0 0

↓

1 1 1 1

(As 0 positive of given Integer should be represented first.)

Two's Complement :

Ex. 1010

Step I : Find One's Complement

0 1 0 1

Step - II : Add 1 to the LSP

0 1 0 1
+ 1
0 1 1 0

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Twos Complement : 0110

Q) 111101

1 1 1 1 0 1

↓

1000010

```
  1 0 0 0 0 1 0
  0 0 0 0 0 0 1
---------------
  1 0 0 0 0 1 1
```

Q) −5

0 1 0 1

↓

1010

```
  1 0 1 0
  0 0 0 1
---------
  1 0 1 1
```

Twos Complement : 1011

Q) −63

0 1 1 1 1 1 1

↓

1000000

2	63
2	31
2	15
2	7
2	3
2	1

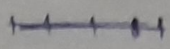
```
  1 0 0 0 0 0 0
            1
---------------
  1 0 0 0 0 0 1
```

Q) −33

One's Complement : 1011110

Two's Complement : 1011111

Sign Bit Representation : 1100001



0100001

↓

1011110

```
 1011110
      01
 -------
 1011111
```

g) +0 , -0

+0 : 0000

↓

0000

```
 1111
 0001
 ----
 0000
```

Twos Complement : 0000

-0 : 0000

↓

1111

```
 1111
 0001
 ----
 0000
```

Twos Complement : 0000

Bitwise NOT :

Ex. int a = 5;

printf("%d", ~a);

Output = -6

~~Short~~

Shortcut : Bitwise NOT of n = -(n+1)

Method : 5 ↘

0101

5 : 0101

~5 : 1010

Two's Complement of 1010 : 0110

1 0 1 0

↓

0 1 0 1

0 1 0 1
0 1 1
0 1 1 0

[Twos Complement of 0110 : 1010]

0110 ⟶ 6

Negative , 6 , ⇒ −6

Q) 11

a = 11
~a

11 : 0 1 0 1 1
11 : 1 0 1 0 0 (One's Complement)

2	11	
2	5	1
2	2	1
2	1	0
	0	1

Two's Complement of 10100 :

(i) One's Complement
(ii) Then Twos Complement

1 0 1 0 0

↓

0 1 0 1 1

0 1 0 1 1
0 0 1 0 1
0 1 1 0 0

Output : −12

Q) a = −6
~a

1001 ⟶ 0110 ⟶ 0111

0110
+ 1
0111

+6 : 0 1 1 0 (Twos Complement Calculation)

↓

−6 : 1 0 1 0

~a : 0101

↳ +5

(i) Do Twos Complement
(ii) Then one's Complement

Shortcut : ~(-n) = +(n-1)

~~(8) Size of~~

~~(8) sizeof Operator~~

(8) Special Operators

(a) sizeof operator

```
int a;
printf("%d", sizeof(a))
```

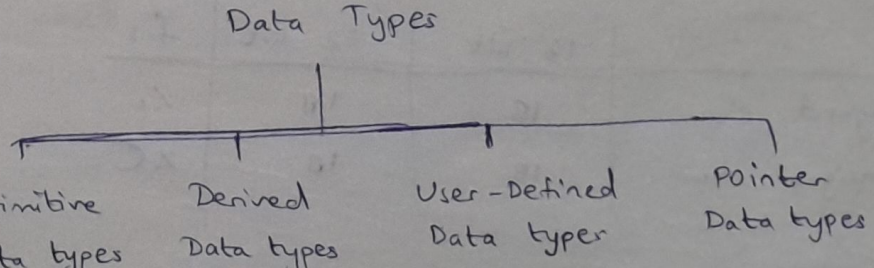
~~Returns the sign of the data~~

Returns the size of the data (constants, variables, .. etc)

(b) comma operator

Link related expressions together

8/9/23 :



16 bit Compiler occupies 2 bytes of memory.

(i) Signed-N — Stores -2^{N-1} to $2^{N-1}-1$

N : No. of bits

~~Short int~~ shortint stores 1 byte memory in a 16 bit compiler

(ii) Unsigned – N bits – Stores 0 to 2^{N-1}

Defaut : ~~un~~signed

	16 bit cc	32 bit cc	f.s
Signed Short Int	1B	2B	%hd
Unsigned short Int	1B	2B	%hu
Signed Int	2B	4B	%d
Unsigned Int	2B	4B	%u
Signed Long Int	4B	4B	%ld
Unsigned Long Int	4B	4B	%lu

B → Bytes

b → bits

1 B = 8 b

Ex. Signed long ~~de~~ int : -2^{32-1} to $2^{32-1}-1$

$\Rightarrow -2^{31}$ to $2^{31}-1$

	16 bit	32 bit	f.s
signed char	1B	1B	%c
Unsigned char	1B	1B	%C

f.s		16 bit cc	32 bit cc	
%f	float	4B	4B	3.4×10^{-38} to 3.4×10^{38}
%lf	double	8B	8B	1.7×10^{-308} to 1.7×10^{308}
%Lf	long double	10B	16B	3.4×10^{-4932} to 3.4×10^{4932}

string → collection of characters
↳ "" ↳ ''

Ex. char ch = 'a';
printf("%c", ch); // a
printf("%d", ch); // 97

Output:
a, 97

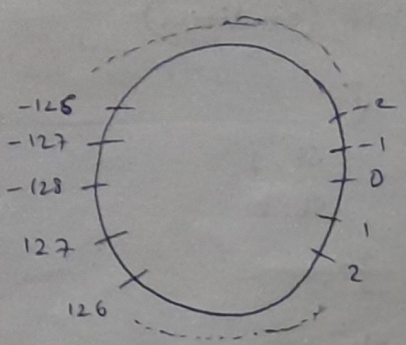
Q) void main()
{
~~unsigned c~~
char ch = 127;
ch = ch + 1;
printf("%d", ch);
}

ch = 127 + 1 = 128

~~Signed char Range : +128~~

Signed char Range : -128 to 127

Output:
~~128~~
-128



→ Conditional ~~Statate~~ Operators:

(i) if (condition)

```
{
    St. 1
    St. 2
    St. 3
}
    St. 4
    St. 5
```

True : St 1, St 2, St 3, St 4, St 5

False : St. 4, St. 5

Ex

```
void main()
int a,b;
printf("Enter Two values \n");
scanf("%d %d", &a, &b);
[redacted]
if (a==b)
    printf("a is equal to b");
printf("End of the program");
```

(ii) if (condition)

```
{
    St. 1
    St. 2
}
else
{
    St 3.
    St. 4
}
```

True : St. 1, St. 2

False : St. 3, St. 4

(9) Read Read Inputs from user and verify which one is larger

```
#include<stdio.h>
void main()
{
   int a, b;
   printf("Enter Two values \n");
   scanf("%d %d", &a, &b);
   if (a>b)
      printf("First number is greater than second number");
   else if (a<b)
      printf("second number is greater than first number");
   else
      printf("Both numbers are equal");
}
```

Method - II :

```
printf("%d is greater than %d", a, b);
```

(iii)

```
if (c1)          // True          // False
{ St.1           St.1
  St.2           St.2
}
else if (c2)     // True          // False
{ St.3           St.3
  St.4           St.4
}
else (c3)        // True          // False
{ St.5           St.5
  St.6 }         St.6
```

(ii)

```
if (C1)                    // True
{ St. 1                    St.1
  St. 2                    St.2
}
else if (C2)               // True
{ St.3                     St. 3
  St.4                     St. 4
}
else if (C3)               // True
{ St.5                     St. 5
  St.6                     St. 6
}
else                       // True
{ St.7                     St.7
  St.8                     St.8
}
```

9) Condition to verify an entered number is positive, negative or equal to 0

```
int a;
printf("Enter a number:");
scanf("%d", &a);
if (a>0)
   printf("Positive");
else if (a<0)
   printf("Negative");
else
   printf("Equal to 0");
```

10/1/25

→ Priority:

(1) Postfix ⇒ () [] → . ++ --

(2) Unary ⇒ ~~~~ ! ~ (type)* & sizeof (Right to Left)

(3) Multiplicative → * / %

(4) Additive → + -

(5) Shift → << >>

(6) Relational → <= < >= >

(7) Equality → == !=

(8) Bitwise AND → &

(9) Bitwise XOR → ^

(10) Bitwise OR → |

(11) Logical AND → &&

(12) Logical OR → ||

(13) Conditional → ?: (right to left)

(14) Assignment → = (right to left)

(15) comma → ,

Q1) printf("value a = %d", (10++));

Output : Error (l value required)

Reason : Post increment can not be performed on constants. It can be only performed on variables.

Q2) int x = 10;

x = (x++) + (++x) + x;

printf("%d", x);

Explanation : $x = \overset{①}{x} + (\overset{①}{x+1}) + (\overset{②}{++x}) + \overset{③}{x}$

12 + 10 + 12 + 12 = 46

Now x = 11 Now x = 12

Output : 46

Q3) int a = 1;
int b = 1;
int c = a || --b;
int d = a-- && --b;
printf("a = %d, b = %d, c = %d, d = %d", a, b, c, d);
return 0;

Explanation:

a → True
∴ c = 1

a-- ⇒ T
Then a = 0

--b ⇒ F
Then b = 0

∴ d = 0

0 → False
Z - {0} → True

Q4) printf("%d", 1 << 2 + 3 << 4);

Explanation:

1 << 5 << 4 (Priority)

32 << 4 (Priority)

32 x 16

Output: 512

```
int main()
{
    return 0;
}
```

Q5) int i = 1, 2, 3;
printf("%d", i);
return 0;

Output: Error

Reason: Compiler thinks 2 as a variable
But Variable cannot start with a number.

Q6) int i = (1,2,3);
printf("%d", i);
return 0;

Output : 3

Reason : Priority from left to right
(Associativity)

Q7) int a = 1;
int b = 0;
b = a++ + a++
printf("%d %d", a, b);
return 0;

int main( )
(aa)
int main(void)

Explanation :

a++ ⟹ 1 and then a = 2
a++ ⟹ 2 and then a = 1

b = 1 + 2 = 3

a = 3

Output :

a = 3
b = 3

Q8) int a = 2, b = 5;

a = a ^ b
b = b ^ a
printf("%d %d", a, b);
return 0;

```
  0 1 0
  1 0 1
-------
  1 1 1
```

Explanation :

Now a = 7
b = 5

```
  1 0 1
  1 1 1
-------
  0 1 0
```

Now a = 7
b = 2

Q9) int i = 12;

int j = sizeof(i++);

printf("%d %d", i, j)

return 0;

Output : i = 12
j = 4

Reason : Operation won't take place in sizeof

Q10) int y=0;

int x = (y!=0);

printf("%d", x);

Output : x = 0
(: False)

Q11) printf("%d", sizeof(printf("jagadeesh")));

return 0;

Output : jagadeesh 4

Reason : No. of charecters in jagadeesh = 9

9 → integer

sizeof(9) = 4

Q12) printf("%d", printf("jagadeesh"));

return 0;

Output : jagadeesh 9

Q13) int i = 3;

printf("%d", (++i)++);

Return 0;

Output : Error

Reason : constant ++ cannot be performed

Q14) int x = 10;

int y = 20;

x += (y += 10);

printf ("%d %d", x, y)

return 0;

Output : x = 40
y = 30

Reason : Firstly, y += 10

⇒ y = 30

∴ x = 10 + 30 = 40

Q15) int a = 10, b = 2, x = 0;

x = a + b * a + 10/2 * a;

printf ("value is = %d", x)

Output: 80

Reason: 10/2 = 5

⇒ a + b x a + 5 x a

⇒ 10 + 2 x 10 + 5 x 10

⇒ 10 + 20 + 50

⇒ 80

Q16) unsigned short var = 'B';

var += 2;

var++

printf ("var: %c, %d", var, var);

Output : E 69

Q17) int x = (20 || 40) && (10);

printf ("x = %d", x);

Output : 1

Q18) int x;

```
x = (printf("AA") || printf("BB"));
printf("%d", x);
printf("\n");
x = (printf("AA") && printf("BB"));
printf("%d", x);
```

Output: AA 2
AABB 1

Q19) int a=3, b=2;

```
a = a == b == 0;
printf("%d %d", a, b);
```

Output : 0 2

Reason : a = a == b == 0

a = a == 0

a = 0

b = 2 (Not modified)

Q20) char val = 250;

int ans;

ans = val + !val + ~val + ++val;

printf("%d", ans)

-128 to 127

Reason : 250 must be stored as -5.

ans = -5 + 0 + 4 + (-4) -

Q21) int i=-1, j=-1, k=0, l=2, m;

m = i++ && j++ && k++ || l++;

printf("%d %d %d %d %d", i, j, k, l, m);

Reason : -1 && -1 && 0 || 2

Output : k = 1
m = 1
i = 0
j = 0
l = 3

→ Nested if :

```
if (C1) :
{
    if (C2) :
    {
      Statement Block 1
    }
    else
    {
      Statement Block 2
    }}
else
{
    Statement Block 3
}
```

→ Switch Statement :

a = 1

```
switch ( a)
{ case 1 :
    St. 1 ;
    St. 2 ;
    break ;
  case 2 :
    St. 3 ;
    St. 4 ;
    break ;
default ;
    St. 5
    St. 6
}
```

15/9/23

Q) In 16 bit 2's complement representation, the decimal number -28 is

Sol: $(28)_{10} \equiv (11100)_2$

2 | 28	
2 | 14	0
2 | 7	0
2 | 3	1
2 | 1	1
0	1

0000000000011100

↓

1111111111100011 → One's complement

1111111111100100 → Two's complement

Q) The 16-bit 2's complement Representation of an integer is 1111 1111 1111 0101, its decimal representation:

Sol: Twos complement of $x = y$

Twos complement of y is x.

0000 0000 0000 1011

Q) Consider the equation $(123)_5 = (X8)_Y$ with X and Y as Unknown. The number of possible solutions is:

Sol: $Y > 8$, $Y \in N$, $X < Y$

$(123)_5 \equiv 25 + 10 + 3$

$= (38)_{10}$

$(X8)_Y = (XY + 8)_{10}$

$\Rightarrow 38 = XY + 8$

$\Rightarrow XY = 30$

30

1	30
2	15
3	10
5	6

$(X,Y) = \{(1,30), (2,15), (3,10), (5,6)\}$

↖ Not possible

$(8 > 6)$

$(X,Y) = \{(1,30), (2,15), (3,10)\}$

3 combinations are possible.

Q)
```
#include<stdio.h>
void main()
{
 if (-1)
 printf("-1");
 else if (0)
 printf("0");
 else
 printf("1");
 }
```

Sol : -1

Q)
```
if (1);
printf("1");
else
printf("0");
```

Sol : Error : Else without if error

Q) Represent $(1234)_{10}$ in base 3.

Sol : $(1200201)_3$

3	1234	
3	411	1
3	137	0
3	45	2
3	15	0
3	5	0
3	1	2
	0	1

Q) Assume two's complement system, what is the decimal equivalent of (111010)

Sol : 000110 → 6

Q) int a, b, c

printf("\n %d", scanf("%d %d", &a &b)

+ scanf("%d, &c));

Sol:

Output : 3

Q) printf("%d, printf("hello") + printf("world"));

Sol:

Output: helloworld10

$(5)_{10}$

$(101)_2$

Q) int x;

printf("%d, %d, %d, %d, %d, %d, x = x+2,

x >> 2, x++, x = x*2, x != 0,

x = 21 << 1*2);

0101
1010
0000
1111

Output: 171, 42, 168, 171, 1, 171

Reason: Assignment always at last.

While printing

$169/4 = 42$

21×2^2

$= 21 \times 4$

$= 84$

Q) int x = 5;

printf("%d", x >> 2);

printf("%d", x)

$5/2^2 = 5/4$

$= 1$

Output: 15

Reason: x >> 2 => Ans = 1

But x's value = 5 only

Q) $(128)_{10} - (1)_{10} - (17)_8 - (2E)_{16}$, Clearly mention Intermediate steps

$(127)_{10} - (17)_8 - (2E)_{16}$

$= (127)_{10} - (15)_{10} - (46)_{10}$

$= (66)_{10}$

Q) The smallest integer that can be represented by a 8-bit number in 2's complement form is

~~11111111~~

Q) printf ("%d \n", sizeof (2.5));
printf ("%d \n", sizeof (2));
printf ("%d", sizeof ('a'));

In a 16 bit compiler

Sol: Output : 8
2
1

Q) printf ("%.%.%.");

Sol: Output : %

Reason: First % → Act as a format specifier
Second %. → Will be printed
Third %. → Act as a format specifier again
Nothing after third %. , ∴ Nothing prints.

Ex. printf ("%.%.%.%.");

Sol : %.%.

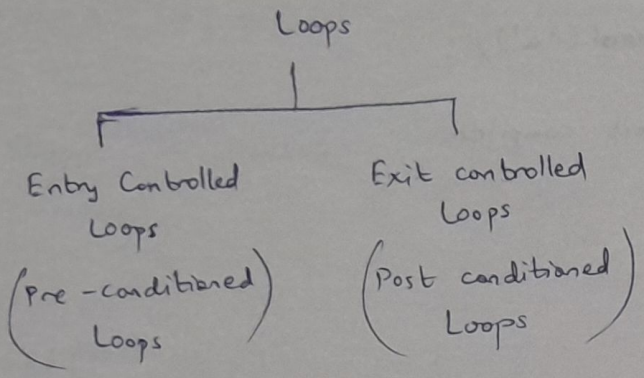
Q) int a;
if (a=2);
printf ("Hi");
else
printf ("Hello");

Sol: Output : Hi
Reason: a=2 is assignment & a=2 gives True i.e. 1.

16/9/23

→ Loops:

- Initialisation
- Termination
- Iteration (Increment / Decrement)



- Verify Condition
- Execute ~~Condi~~ Block of statements Until condition becomes false

Ex. For Loop, While Loop

- Execute block of statements atleast once
- Verify Condition

Ex. Do while loop

(i) While Loop:

```
Initialization
While (condition)
{
    St. 1;
    St. 2;
    iteration
}
```

Ex.
```
i = 0;
while (i<5)
{ printf("%d \n", i);
  i=i+1
}
```

Output:
0
1
2
3
4
5

Q) Write a C program to print numbers from 5 to 1 using while loop.

Sol:

```
i = 5
while (i > 0)
{
    printf("%d \n", i);
    i = i - 1;
}
```

Q) Write a C program to compute the individual digits sum of a given number.

Sol:

```
printf(  );
scanf(  );
int sum = 0;
while (n != 0)
{
    rem = n % 10;
    sum = sum + rem;
    n = n/10;
}
printf("%d", sum);
```

Q) Reverse a number using while loop

H.W

```
int rev = 0
while (n != 0)
{
    rem = n % 10;
    rev = rev x 10 + rem;
    n = n/10;
}
printf("%d", rev);
```

Q) Write a C program to verify whether given number is Armstrong number.

Ex. 123

$123 == 1^3 + 2^3 + 3^3$ ✗

only then it is a Armstrong no.

Ex. 2145 $= 2^4 + 1^4 + 4^4 + 5^4$ ✗

Ex. 153 $= 1^3 + 5^3 + 3^3$ ✓

153 → Armstrong no.

```
int n;
while(n!=0)
{
    on = n;
    int num = 0;
    while (n!=0)
    {
      c=c+1;
      n=n/10;
    }
```

— Finding no. of digits

```
    n = on;
    int sum = 0;
    while (n!=0)
    { rem = n%10;
      sum = sum + pow(rem, c);
      n = n/10;
    }
    n = on;
```

— Summation of powers

```
if (sum == "on")
  printf("Armstrong no.");
else
  printf("Not");
```

→ For Loop :

```
for (init. ; cond ; iteration)
{


}
```

Ex.
```
int i;
for (i=1 ; i<6 ; i++)
{
  printf("The i value is %d \n", i);
}
```
(① ② ④ ③)

→ Initialisation will be done only once.

→ After ④, ② will be executed

① → ② → ③ → ④ → ② → ③ → ④

Alternate Method :

```
int i=0;
for ( ; i<=5 ; )
{
  printf("The i value is %d \n", i);
  i++;
}
```

Q) Write a C program to print numbers from 5 to 1 using for loop.

```
for (i=5 ; i>0 ; i--)
{
  printf("%d\n", i);
}
```

9) Take 3 Inputs from the user,

which number of multiplication table,

Starting from what, ending with what,

And then display Multiplication table

```
#include<stdio.h>
void main( )
{
[redacted],
int st, en, num, i, x ;
printf ("Enter a number : ") ;
scanf ("%d", &num) ;
printf ("Enter starting point : ");
scanf ("%d", &st );
printf ("Enter Ending Point : ");
scanf("%d", &en);
for (i = st ; i <= en ; i++)
{
  x = i x num ;
  printf ("%d x %d = %d", num, i, x) ;
}
}
```

(OR)

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

→ Break and Continue Statements

→ Break Syntax :

break ;

Ex.

```
int i ;
for (i=1 ; i<=5 ; i++)
{
  if (i == 3)
    break ;
  else
    printf ("%d \n", i);
}
```

→ Continue syntax:

```
continue;
```

↳ Leave the current cycle of the loop & starts with the next loop

Ex.
```
int i;
for (i=1; i<=5, i++)
{
  if (i==3)
  {
    continue;
    printf("%d \n", i);
  }
  else
    printf("%d \n", i);
}
```

Continue skips the statements after continue statement and continues the next iteration.

Output :
```
1
2
4
5
```

(9)
```
#include<stdio.h>
int main()
{ int i;
  if (printf("0"))
     i=3;
  else
     i=5;
  printf("%d", i);
  return 0;
}
```

Output : 03

Q) Int i = 2;

```
switch (i)
{
   case 0 : printf("Hi"); break;
   case 1 : printf("How are you"); break;
   default : printf("I am fine");
}
```

Output : I am fine

Q) Int a = 5;

```
switch (a)
{
   default : a = 4;
   case 6 : a--;
   case 5 : a = a+1;
   case 1 : a = a-1;
}
printf("%d \n", a)
return 0;
```

Output : 5

Reason : a = 5+1 = 6

a = 6-1 = 5

Q) char check = 'a';

```
switch (check)
{ case 'a' || 1 : printf("Geeks");
   case 'b' || 2 : printf("Quiz");
                   break;
   default : printf("GeeksQuiz")
}
```

Output : Syntax error

(Logical Or)

Q)
```
int si=1;
switch(++si - si++)
{
  case 1:
  printf("First");
  break;
  case 2:
    printf("second");
    break;
  default:
    printf("Bye");
    break;
}
```

Output : First

If 'int' is replaced by 'short int',

Output : Bye

Q)
```
int i=0;
for(i=0; i<20; i++)
{
   switch(i)
   {
      case 0: i+=5;
      case 1: i+=2;
      case 5: i+=5;
      default: i+=4;
      break;
   }
   printf("%d", i);
}
```

Reason:

i=5

5+2=7

7+5=12

12+4=16

[i=16]

i → 17

17+4=21

[i=21]

22 > 21 → Ends

Output : 16 21

Q) #define ODD 1

```
#define EVEN 0

void main()
{ int i=3
  switch(i%2)
  {
    case EVEN : printf("Even");
                break;
    case ODD  : printf("Odd");
                break;
```

Q)

```
int i=1;
for (i=0; i<10; i+3)
{
  printf("Hi");
}
```

Syntax : i+3
↓
Hi will be printed infinitely many times as i=0 only.

If 'i+3' is replaced with 'i=i+3',

Output : hihihihi

Q)

```
int i=1024;
for ( ; ; i>>=1)
printf("Hi");
```

i>>=1
⇓
i=i>>1

Q)

```
int n;
for (n=9; n!=0; n--)
  printf("n=%d", n--);
```

n ⟶ -1

Output : Infinite Loop

Q)

```
int i;
for (i=1; i!=10; i+=2)
printf ("Hi")
```

Output : Infinite Loop

Q) int i=0;

```
for (printf("1st \n"); i<2 && printf("2nd\n");
        ++i && printf("3rd \n"))
{
    printf("*\n");
}
```

Output:

1st (i=0)
2nd
*
3rd (i=1)
2nd
*
3rd (i=2)

Q) int i = -5;

```
while (i<=5)
{
    if (i>=0);
        break;
    else
    {
        i++;
        continue;
    }
    printf("GeeksQuiz");
}
```

Output:

No output

if → True,
Then break out of loop

else → True,
Then continue out of loop

Q) int i=3;

```
while (i--)
{
    int i=100;
    i--;
    printf("%d,", i);
}
```

Output:

99,99,99,

Reason:

0 → false
(if i=0)

→ Do-while loop :

```
Variable Initialisation;
do
  {
    statements;
    incre/decre;
  }
while (condition);
```

Ex.
```
void main()
{
  int i=1;  ①
  do
          ② ⑤ ⑧
  { printf("%d", i);
    i++;  ③ ⑥ ⑨
  }
        ④ ⑦ ⑩
  while (i<=5);
}
```

Output : 1,2,3,4,5,

Ex
```
int i=5;
do
{ printf("%d", i);
  i--;
}
while (i>0);
```

Output : 54321

→ Nested Loops :

→ Nested For Loop :

```
for ( initialisation ; condition ; incre/decre)
{ for ( initialisation ; condition ; incre/decre. )
  { statement;
  }
}
```

Ex.
```
#include<stdio.h>
void main()
{
  for(int i=1; i<=3, i++)
  {
    for(int j=1, j<=i, j++)
    {
      printf("%d", i);
    }
  }
}
```

Inner for loop will be initialized again.

Output:

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

(g)
```
#include<stdio.h>
void main()
int i;
{
  for(i=1; i<=2; i++)
  {
    printf("Inside Outer Loop\n");
    for(int j=0; j<=1; j++)
    {
      printf("Inside Inner Loop\n");
      printf("%d %d \n", i, j);
    }
  }
}
```

Output:

```
Inside Outer Loop (i=1)
Inside Inner Loop
1 0
Inside Inner Loop
1 1
Inside Outer loop
Inside Inner loop
2 0
Inside Inner Loop
2 1
```

→ Arrays:

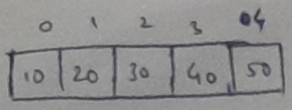
↳ Collection of Homogenous elements

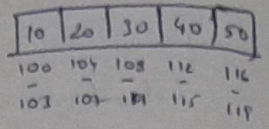
Syntax:

datatype arrayname[size];

Ex. int a[10];

Ex. int a[5] = {10, 20, 30, 40, 50};





index = a a[index]

printf("%d \n", a[2]);

a[2]

100 + 8 2 x 4

= 108

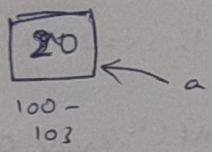
→ Pointer is a variable which stores the address of another variable.

printf("%d", &a) → Prints Memory location

↓

prints 100.

int a=20;

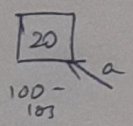


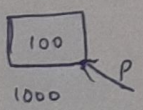
Syntax :

datatype *pointername ;

Ex. int *p ; int a=20;

p = &a ;





a → 20

&a → 100

p → 100

*(p) → 20

&p → 1000

*(&p) → 100

} printf("%d") Commands

→ Two dimensional Array :

datatype arrayname [size1] [size2]

Ex. int a[3][3]

00	01	02
10	11	12
20	21	22

printf ("%d", a[2][1])

Ex. int a[2][2] = {1, 2, 3, 4}

00	01
1	2
10	11
3	4

(OR)

int b[3][2] = {{1, 2}, {3, 4}, {5, 0}}

1	2
3	4
5	0

Q) Write a C program to write a matrix addition

~~int~~

~~int a, b, c, d;~~

~~int array a [a][b];~~

```
int a[10][10], b[0][10], r, i, j, c, sum[10][0];
printf("row no : ");
scanf("%d", &r);
printf("column no. : ");
scanf("%d", &c);
```

```
for (i=0 ; i<r ; i++)
{
    for ( j=0 ; j<c ; j++)
    {
        scanf ("%d", &a[i][j]);
    }
}

for (i=0 ; i<r ; i++)
{
    for (j=0 ; j<c ; j++)
    {
        scanf ("%d", b[i][j]);
    }
}

for (i=0 ; i<r ; i++)
{
    for (j=0 ; j<c ; j++)
    {
        sum[i][j] = a[i][j] + b[i][j] ;
    }
}

for (i=0 ; i<r ; i++)
{
    for (j=0 ; j<c ; j++)
    {
        printf ("%d ", sum[i][j]);
    }
if (j == c-1)
    printf ("\n");
}
```

(g) Write a C program for Matrix Transposing

```
int i,j, a[10][10], t[10][10];
printf("rows no. & columns no.");
scanf("%d %d", &r &c);

for (i=0; i<r; i++)
 {
    for (j=0; j<c; j++)
    {
       scanf("%d", &a[i][j]);
    }
}

for (i=0; i<r; i++)
{
   for (j=0; j<c; j++)
   {
     printf("%d", a[j][i]);
   }
  if (i==c-1)
    printf("\n");
}
```

00 01
1 2

010 11
3 4

→ Multidimensional array:

datatype arrayname [size1][size2][size3];

→ Functions :

- User - Defined functions
- Pre - Defined functions

Syntax :

datatype function name (dt1, dt2) ;

Ex. int fun (int, int) ;
int fun (int a, int b) ;

Ex int fun (int a, int b);
{ St. 1
St. 2
St. 3
return integer literal ; }

int s = fun (10, 20) ;

Q) Write a C programe to compute sum of two numbers using functions.

```
#include<stdio.h>
int addfn (int, int);
void main( ) {
  int c,d, sum
  printf ("Enter two values");
  scanf ("%d %d", &c, &d);
  sum = fun(c, d);
  printf("The result is %d", sum); }
int fun(int a, int b)
{ int c = a+b;
  return c; }
```

P) Compute avg. of 3 numbers using a C program.

```
#include<stdio.h>
int avgfun(int, int, int);
void main()
{
    int a,b,c,avg;
    printf("Enter 3 numbers : \n");
    scanf("%d %d %d", &a, &b, &c);
    avg = avgfun(a,b,c);
    printf("The average is %d", avg); }
int avgfun(int a, int b, int c)
{   int e = (a+b+c)/3;
    return e; }
```

Forms of User-Defined functions.

(1) No Return type with no parametre
(2) No return type with parametre
(3) Return type with no parameter
(4) Return type with parameter

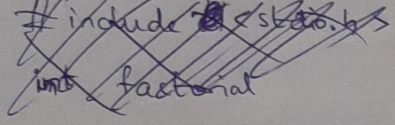
Recursion:

Function Calling Itself

→ Factorial :

```
int factorial(int n)
{
    if (n==0 || n==1)
        return n;                    // base case : gives solution to some part of the problem
    else
        return n* factorial(n-1);    // general case : Minimising the code
}
```

g) Write a program to find factorial of a number using Recursive function :



```
#include <stdio.h>
int factorialfn(int n);
void main() {
    int fact, x;
    printf("Enter a Number : ");
    scanf("%d", &x);
    fact = factorialfn(x);
    printf("The factorial of %d is %d", x, fact); }

int factorialfn (int n) {
    if ((n==0)||(n==1))
        return 1;
    else
        return n* factorialfn(n-1); }
```

Q) Write a C program to display first n numbers in fibonacci series using Recursive Function.

```
#include<stdio.h>
int fibonacci(int x, int y);
void main() {
    int a,b,i;
    printf("Enter first two numbers : ");
    scanf("%d %d", a,b);
    i = fibonacci(a,b); }
int fibonacci(int x, int y) {
```

→

```
char a[20];
printf("Enter");
scanf("%s", a);
printf("%s", a);
```

Input : Harith ↵
Output : Harith

(OR)

```
char a[20];
puts("Enter");
gets(a);
puts(a);
```

Input : Harith ↵
Output : Harith ↵

→ strlen() : Computes string length
strcpy() : Copy a string to another
strcat() : Concatenates (joins) two strings
strcmp() : Compares two strings
strlwr() : Converts string to lowercase
strupr() : Converts string to Uppercase

In 'string.h' library.

Q) #include <stdio.h>
int main() {
int a = 2;
switch (a) {
}
printf("Know Program");
return 0;
}

Output : Know Program
Reason : No error.
Only statements are not present so switch case wont execute.

Q) int a;
switch (a)
{
printf("Hello");
}
printf("Know Program");

Output : Know Program
Reason : No case so no statement will be executed inside switch block.

Q) int a = 50;
switch (a) {
Case a<50 : printf("x"); break;
Case a>=50 : printf("y"); break;
default : printf("z"); break; }
printf("\n"); }

Output : Error
Reason : Case label should be Integer.

Q) switch (printf("HelloWorld\n"))
{
Case 10 : printf("hi"); break;
Case 11 : printf("How are you"); break;
default : printf("time waste");
}

Output :
HelloWorld
How are you
Reason :
\n also is considered as a charecter

Q) int k;
for (k=0;;)
printf("hi");

Output: hihihihihihi..... ∞
Reason: No error because syntax is correct. But no iteration so ∞.

Q) int arr[] = {2,3,4,5,6,7};
printf("%d", arr[2]);
printf("%d", arr[2*2]);

Output: 46
Reason: 2x2=4 will be evaluated.

Q) int a[10] = {1,2,3,4,5,6,7,8,9};
printf("%d", ++a[0]);

Output: 2
Reason: ++1=2

Q) int arr[4] = {3,4,5,6};
int k[4];
k = arr;
printf("%d\n", k[1]);

Output: Error
Reason: k variable cannot store

Q) int arr[5];
// Assume base address of arr is 2000 and size of integer
// is 32 bit.
arr++;
printf("%u", arr);

Output: Error
Reason: & value required a++, a should be integer

Q) int arr[5];
// Assume base address of arr is 2000 and size of integer is
// 32 bit.
printf("%u %u", arr+1, &arr+1);

Output: 2004 2020
Reason: 2000+4(1)
2000+4(5)

Q)
```
int i;
int arr[5] = {1};
for (i=0; i<5; i++)
    printf("%d ", arr[i]);
```

Output:

1 0 0 0 0

Reason:
Default value is 0.

Q)
```
int i=0;
for (; i<5; i++);
    printf("%d", i);
printf("%d", i);
```

Output: 5

Reason: As semicolon beside 'for' statement, statements inside for loop does not get executed but iteration and checking condition get executed until condition is false

Changing value of a,

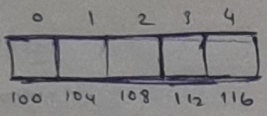
$a = 10$

(OR)

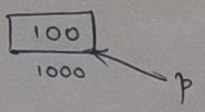
$*p = 10$

→ Arithmetic operations on pointers: (32 bit cc)

```
int a[5];
```




```
int *p;
p = &a[0];
```

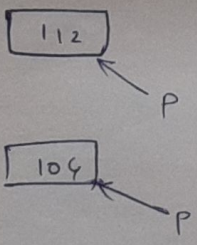



```
p = p+3;
```

$100 + 3 \times 4$
$= 112$
↳ a[3]

```
p = p-2;
```

$112 - 2 \times 4$
$= 104$




```
int *p; p = &a[0];
int *q;
q = &a[3]
```

```
q-p;
```
↳ $\frac{112-100}{4} = \frac{12}{4} = (3)$

```
q+p;
```
↳ Not possible

Pointers cannot be added, Multiplied or Divided.

Ex. int *p;

p = &a[0];

p++;

→ p = p+1;

100 + 1 × 4 = 104

p--;

→ p = p-1

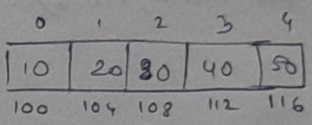
104 - 1 × 4 = 100

Ex. int *q;

q = &a[1];

int *p;

p = &a[0];



*p + *q

→ 10 + 20 = 30

*100 = 10

*104 = 20

*p - *q

→ 10 - 20 = -10

(*p) * (*q)

→ 10 × 20 = 200

(*p) / (*q)

→ 10/20 = 0

* → Call by Value and Call by Reference in Functions :

Call by Value :

Ex.
```
#include <stdio.h>
void add(int, int);

void main();
{
  int c=10;
  int d=15;
  add(c,d);
  printf("%d %d", c, d);
}

void add( int a, int b)
{
  a = a+5;
  b = b+10;
  Printf("%d %d", a, b);
}
```

Output:

Tracing:

c = c+5 = 10+5 = 15

d = 15+10 = 25

Output:

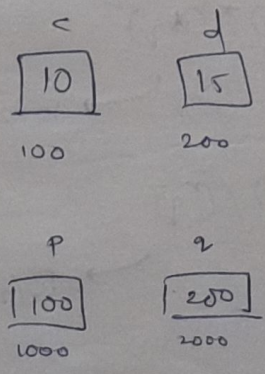
15 25 10 15

Reason:

Operating on formal Parametres won't affect Actual Parameters.

Ex.
```
#include <stdio.h>
        :
        {
        :
        }
        :
        :
  add(&c, &d);
        :
        :
        :
        :
void add(int *a, int *b)
{
  *p = *p+5;
  *q = *q+5;
  printf("%d %d", *p, *q);
}
```

Tracing:



Output:

15 25 15 25

Reason:

Now it affects actual.

→ Pointer is of two types:

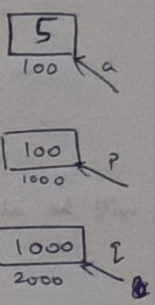
(1) Typed

(2)

```
int m=5, n=10, a,b,c,d;
int *p1, *p2, *p3;
p1 = &m;
p2 = &n;
a = (*p1) + (*p2);   //a=15
b = (*p1) - (*p2);   //b=-5
c = (*p1) * (*p2);   //c=50
d = (*p1) / (*p2);   //d=0
```

→ Pointer to Pointer:

```
int a=5;
int *p;
int **q;
p = &a;
q = &p;
```




```
printf("%d", a);
printf("%d", *p);
printf("%d", **q);
```

} → 5

**q
= **(1000)
= *100
= 5

&q → 2000
*q → 100

→ Types of Pointers:

(i) Null Pointer (int *p = null;)

(ii) Void Pointer (void *p)

↳ Generic Pointer

(iii) Wild Pointers

↳ No Initialization of Pointer (No address value)

↳ Random Pointers

↳ Dangling Pointers

STRUCTURE :

Syntax :

```
struct structurename
{
    datatype member1;
    datatype member2;
    datatype member3;
};
```

Memory will not be allocated.

Structure → User Defined Datatype

Declaring Variables for Structures :

```
struct structname v1, v2;
```

(OR)

```
struct structname
{
    ;        // 4B
    ;        // 4B
             // 1B
} v1, v2;    // 9B memory will be alloted.
```

Declaration of variable inside structure ✓

Initialization of variable inside structure ✗

→ Compilation error

```
struct structname
{
    int x=0;
    int y=0;
};
```

Wrong (Error)

Assigning :

```
struct structname variablename = {list of items};
```

a/11/23

Ex.

```
struct Student
{
    int rollno;
    float cgpa;
};
void main()
{
    struct student s1 {1, 9.5};
    printf("%d", s1.rollno);
    printf("%f", s1.cgpa);
}
```

(or)

```
struct student
{
    int rollno;
    float cgpa;
} s = {56, 8.2};
void main()
{
    .
    .
    .
}
```

Q) Input from users.

A)

```
#include<stdio.h>
#include<string.h>
struct Student
{
    int rno;
    char name[20];
    float cgpa;
};
void main()
{
    int b; char a[20]; float c;
    printf("Enter Name : \n"); scanf("%s", a)
    printf("Enter Roll No. : \n"); scanf("%d", &b);
    printf("Enter CGPA : \n"); scanf("%f", &c);
    struct Student s1 {b, a, c};
    printf("Name : ", s1.name);
    printf("Roll No. : ", s1.rno);
    printf("CGPA : ", s1.cgpa);
}
```

→ Array of Structure :

Ex. struct Employee
{ int id ;
char name[5];
float salary ;
} emp1 , emp2 , emp3 ;

Ex. struct Employee
{
int id;
char name[5] ;
float salary ;
} e[3];

Q) Input from User

```
#include<stdio.h>
#include<string.h>
struct Employee
{
  int id ;
  char name[5];
  float salary ;
} emp[3] ;
void main( )
{
  int i,n ;
  for (i=0 ; i<3 ; i++) {
    printf("Enter ID of Emp %d :", i+1);
    scanf("%d", &emp[i].id) ; }
  for (i=0; i<3 ; i++) {
    printf("Enter Name of Emp %d", i+1);
    scanf("%s", emp[i].name); }
  for (i=0 ; i<3 ; i++){
    printf("Enter salary of emp %d", i+1);
    scanf("%d", &emp[i].salary); }
  printf("Enter Employee number to
        show data of that employee : ");
  scanf("%d", &n);
  printf("Name : %s \n ID : %d \n
        Salary : %d \n", emp[n].name,
        emp[n].id , emp[n].salary);
```

→
```
#include <string.h>
#include <stdio.h>
struct student
{ int rno;
  char name[10];
};
void show(struct student s2);
int main()
{
  struct student s1
  s1.rno = 111;
  strcpy(s1.name, "Akhil");
  show(s1);
  return 0;
}
void show (struct student s2)
{
  printf("Rno: %d", s2.rno);
  printf("Name: %s", s2.name);
}
```

14/11/23

```
#include<stdio.h>
void display (int, int);
void main ( )
{
    int a[5] = {1,2,3,4,5}
    display (a[0], a[2]);
}
void display (int b, int c)
{
    int result = b+c;
    printf("%d", result);
}
```

[OR]

```
#include<stdio.h>
void display (int a[]);
void main ( )
{
    .
    .
    .
display (a);
}
void display (int b[])
{
  int i;
  for (i=0; i<5; i++)
     printf("%d \n", b[i]);
}
```

Q) Write Program to input array and find largest element

```
#include<stdio.h>
void largest_fn(int) ;
void main( )
{
   int a[1000];
   for (i=0; i<1000; i++) {
      scanf("%d", &a[i]);
      if (a[i] == '\n')
         break; }
   int l;
   for (l=0; a[l] != '\0'; l++);
   largest_fn (a, l)
}
```

```
void largest_fn (int b[], l)
{
  int i; int max = b[0];
  for (i=0; b[i] != '\0'; i++) {
     if (b[i+1] > b[i])
        max = b[i+1]; }
}
```

Syntax:

(i) struct structurename *pointername;

(Introducing)

(ii) pointername = &structure's variablename;

(Accessing)

(iii) pointername -> data member name;

(Printing)

Ex.

```
#include<stdio.h>

struct books
{
    char subject[100];
    int book_id;
};

void show(struct books *book2);
int main()
{
  struct books book1;
  strcpy(book1.subject, "C programming");
  book1.book_id = 111
  show(&book1);
  return 0;
}
void show(struct books *book2);
{
  printf("Subject: %s", book2->subject);
  printf("Book ID: %d", book2->book_id);
}
```

→ Nested structures :

```
struct student
{
    char name[15];
    int age;
    struct address
    {
        char city[50];
        int pincode;
    } add;
} st;
```

[OR]

```
struct address
{
    char city[50];
    int pincode;
};
struct student
{
    struct address add;
    char name[15];
    int age;
} st;
```

```
strcpy(st.add.city, "vizag");
st.add.pincode = 123;
```

→ Union :

↳ Same as structure

Difference : Union declares the highest storage memser as its memory location

Ex.
```
Union data
{
    int i;
    float d;
    char name[20];
} d1;
```

Here storage : int → 4
float → 4
Chars → 20 × 1 = 20

Hence,

Storage Alloted = 20

→
```
#include<stdio.h>
union student{
    int rollno;
    char name[20];
    float cgpa;}
void main() {
```

```
union student S1;
printf("Enter the name: ");
scanf("%s", &S1.name);
printf("Enter rollno: ");
scanf("%d", &S1.rollno);
printf("Enter cgpa: ");
scanf("%f", &S1.cgpa); }
```

→ In structure, we cannot declare variable based. Memory cannot be allocated.

```
struct side {
    char name[] = "Ravi";   ✗
    int noofpages = 200;
};
```

Ex.
```
struct {
    int i;
    char c;
} myvar = { i=100, c='A' };
printf("%d %c", myvar.i1, myvar.i2);
return 0; }
```

20/11/23
9)

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    FILE *p;
    char name[50];
    p = fopen("file1.txt", "w");
    if (p == Null) {
        printf("file is not opened \n");
        exit(1); }
    else {
        printf("Enter the name:");
        scanf("%s", name);
        fprintf(p, "%s", name);
        fclose(p); }
}
```

9) Write C program which takes n-student's information and each students' name and Roll no.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    FILE *p;
    char name[50];
    int rollno; int n;
    p = fopen("file1.txt", "w");
    if (p == Null) {
        printf("file not opened \n");
        exit(1); }
    else {
        printf("Enter no. of students:");
        scanf("%d", &n); }
        for (i=0; i<n; i++)
        { printf("Enter Roll no. & name");
          scanf("%d %s", &rollno, name);
          fprintf(p, "name = %s,
                  rollno. = %d", name,
                  rollno);
        }
    fclose(p); }}
```

Q) Write a C program to read the contents of a file.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    FILE *p;
    char name[50];
    p = fopen("file1.txt", "r");
    if (p==Null) {
        printf("file Not found \n");
        exit(1); }
    else {
        fscanf(p, "%s", name);
        printf("%s", name);
        fclose(p); }}
```

If Replacing %d with
fscanf(p, "%d", &n)

→ Reading and Writing Text file:

① fprintf: Inserting data into a text file.

Syntax:

fprintf(file pointer name, "format specifier", Name of Variable);

② fscanf: Reading (or Accessing ~~into file~~ from a text file.

Syntax:

fscanf(file Pointer name, "format specifier", Address of Variable);

③ fputs: Inserting data into text file

Syntax:

fputs(char arrayname, file pointername);

④ fgets: Reading (accessing) data from a text file

Syntax:

fgets(char arrayname, int n, file pointer name);

[n : how many characters need to be read]

↳ Ex. FILE *pf;
char a[200];
pf = fopen("data4.txt", "r");
fgets(a, 200, pf);
printf("%s", a);
fclose(pf);}

} 2n ~~void~~ main function

Note :

fcloseall; ⟶ closes all text files

↳ Instead of fclose(pf);

2/11/23

Q) Write a C program to copy ~~one~~ contents of one text file into another file.

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
  FILE *p, *q; char c;
  p = fopen("file1.txt", "r");
  q = fopen("file2.txt", "w");
  if((p==NULL) || (q==NULL))
  { printf("File not found \n");
    exit(1); }
  else {
    while((c=fgetc(p)) != EOF)
    {
      fputc(c,q);
    }
    fcloseall; }

/* FILE *r;
r=fopen("file2.txt", "r");
while((c=fgetc(r)) != EOF)
{
  printf("%c", c);
}
fclose(r); */
```

→ Random Access fuactions.

① fseek()

∴ → move a file pointer to a specific location

Syntax:

fseek(File Pointer, offset, Position);

Position can have 3 values:

0 — SEEK_SET — Beginning of file
1 — SEEK_CUR — Current position
2 — SEEK_END — End of file

Offset

↳ +ve or -ve integer
↳ No. of bytes to be skipped.

② ftell()

→ provide current position of the pointer in the file.

∴ Count from the beginning of file.

Syntax:

int variable = ftell(fptr);

③ rewind()

→ to move the file pointer to the beginning

Syntax:

rewind(fptr);

↳ Replacement "fseek(p,0,0);"

28/11/23

→ Dynamic Memory Allocation

(i) malloc (memory allocation)

Syntax : (void *) malloc (size) ;

```
int *p = (int *) malloc (10 x sizeof(int)) ;
```

Ex.
```
#include<stdio.h>
#include<stdlib.h>
void main()
{
  int n, i ;
  printf("Enter the n value : \n");
  scanf("%d", &n);
  int *p = (int *) malloc (n x sizeof(int))
  if (p == NULL) {
     printf("Memory not allocated");
     exit(1);
  }
  else {
     for (i=0 ; i<n ; i++) {
        printf("Enter the elements : ");
        scanf("%d", p+i) ;
     }
     for (i=0; i<n ; i++)
        printf("%d", *(p+i));
  }
}
```

```
int a ;
scanf("%d", &a);
int *p ;
scanf("%d", &*p);
("%d", p);
```

5) Allocate the memory dynamically & compute the sum of elements.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int n, i, sum = 0;
    printf("Enter the n value:");
    scanf("%d", &n);
    int *p = (int *)malloc(n * sizeof(int));
    if (p == NULL) {
        printf("Memory Not Allocated");
        exit(1); }
    else {
        for (i=0; i<n; i++) {
            printf("Enter the elements:");
            scanf("%d", p+i); }
        for (i=0; i<n; i++) {
            sum = sum + *(p+i);
            printf("Sum is :%d", sum); }
        free(p);
        p = NULL; }
}
```

free(p) → Deallocating the memory of p.

(ii) calloc : (Continuous allocation)

Syntax:

(void *) calloc(n, sizeof(n))

n → no. of blocks need to be created.

sizeof(n) → size of each block.

Ex. void main()

{

display();
display();
display();

}

Output: 1
1
1

void display() {

int x = 0;
x++;
printf("%d", x);

→ Storage Classes

- Automatic
- static
- Register
- External

Syntax:

Scope datatype variablename;

default scope: Automatic [i.e. Local variable]

(i) Ex. auto int a = 3;

default value: Garbage Value
Storage Place: RAM Memory
Initial Value: Garbage Value
Scope: Local to Block
Lifetime: Within the block

(ii) Ex. static int a = 4;

default value: 0
Storage Place: RAM Memory
Initial value: 0
Scope: Local to Block
Lifetime: End of Program

Ex. void main( ) {

```
    d( );
    d( );
    d( ); }

Void d( );
    static int x=0;
    x=x+1;
    printf("%d", x);
```

Output:

1
2
3

Static Variable:

Initializes only once & remains into existance until end of Program.

(iii) Register Variable.

Ex Register int a=5;

Register variable tells compiler to store variables in CPU register rather than Memory

Storage Place: register memory
Default Value: garbage value
Scope: Local to Block
Lifetime: Within the block

(iv) External

Ex, extern int a = 5;

Ex:

file1.c

```
#include<stdio.h>
int a;
void fun(int);
void main()
{
  fun()
  {
     a = a+9;
     printf("%d", a);
  };
}
```

file2.c

```
#include "file1.c"
extern int a;
void main(){
  fun();
}
```

Global Variable — Can be accessed only within the program

External Variable — Can be accessed from other files

~~Storage~~ (Default value = 0)

→ Type casting (or) Type conversion :

- Implicit typecasting : Conversion done by compiler
- Explicit typecasting : Conversion done by programmer

↳ Syntax :

(datatype) variablename

Ex. char c = 'a';
int b = (float) c