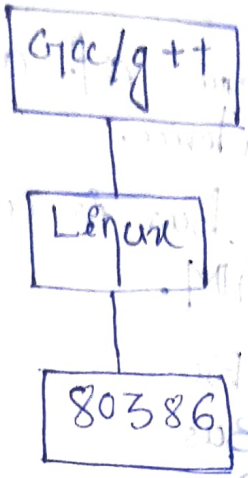
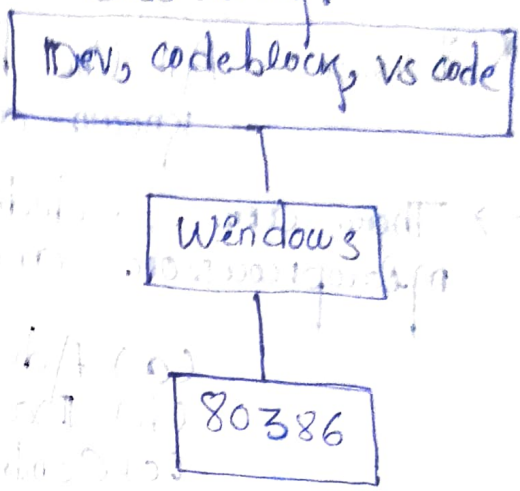
List of OS

MSDOS
Linux
windows
MAC

List of compelere

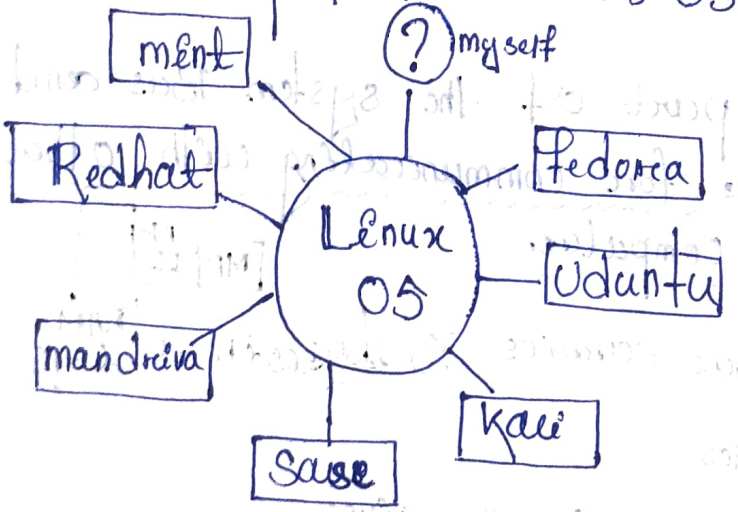
Turbo c/c++
Borland c/c++
Vs code
Dev c/c++
gcc/g++
Code block.





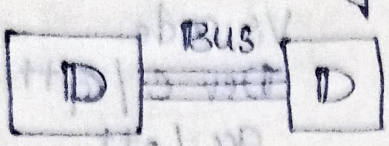
18.04.2022

* Microsoft developed the windows OS.



* Linux operating system is an opensource
* Python, Java, php, linux, Mysql etc all are open source
* Windows, Oracle are licence source

Bus:— Bus is a set of wires connected from one device to another device of a system.



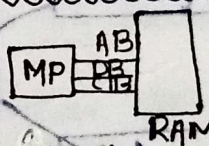
one device to another device different bus wire connected is known as Bus

* **Device:—** All the hardware of a system is known as a device.

→ There are 3 dedicated bus is connected between microprocessor and RAM.

(a) Address Bus
(b) Data Bus
(c) Control Bus.

* Address Bus allows the cpu to send the address to RAM
* Data Bus allows the actual data transfer to the CPU
* Control Bus is part of the system Bus and is used by CPUs for communicating with other devices either the computer.



→ If address bus carries 3 wires.

$2^3 = 8$ bits

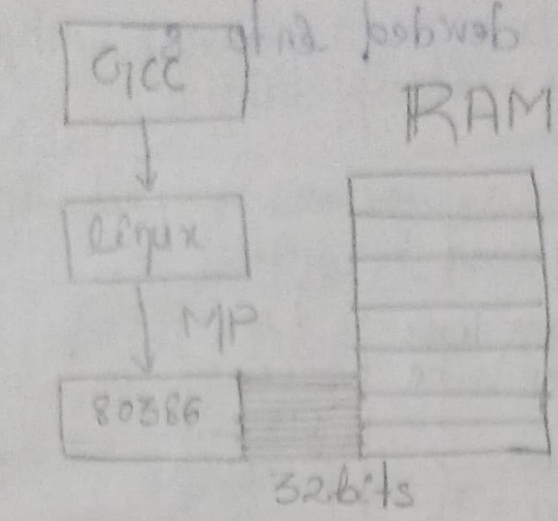
→ If address bus carries 5 wires

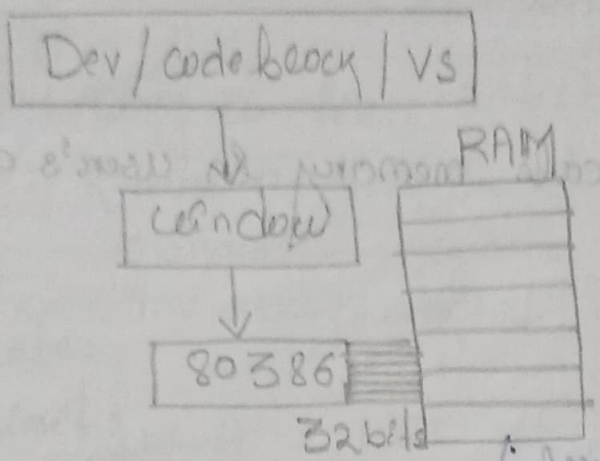
$2^5 = 32$ bits

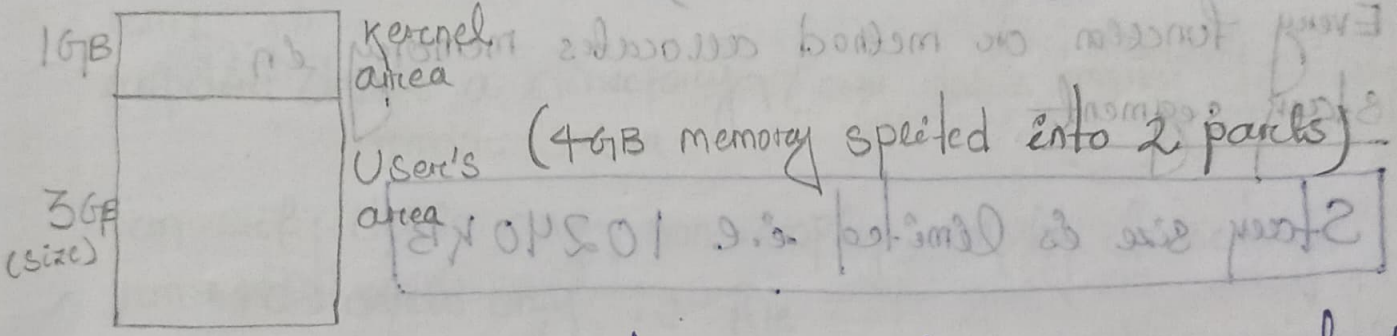
→ 80386 microprocessor carries 32 bits address Bus and hold 4 GB of space (Memory)

$2^{32} = 4294967296$
$= 4GB$

1024 KB = 1 MB





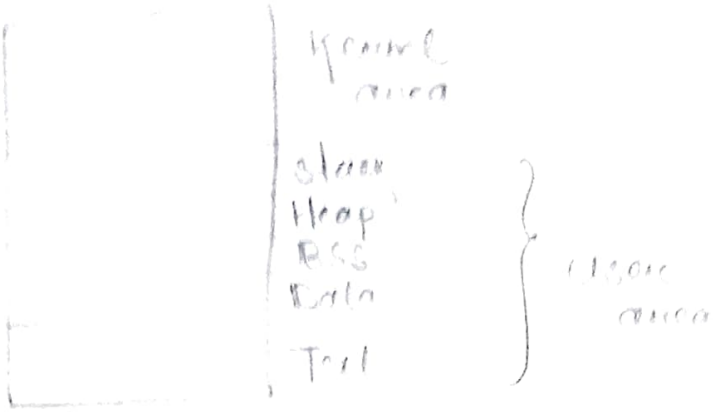


→ Every program allocate memory in user area, but the device drivers program allocate memory in Kernel area

→ C program can be written in Kernel area

- Device Drivers → A device driver is a computer program that operates or controls a particular type of device that is attached to a computer.

* User. area of memory is devided into 8 partition.



* Main() is a function
* The main function allocate memory in user's area
* Main(); Recursion

```
main()
printf("Hi");
main(); (segmentation fault)
```

* Every function or method allocates memory in stack segment

Stack size is limited i.e 10240 kB

```
main()
{
malloc();
calloc();
realloc();
```

These functions allocates memory in heap segment

BASIC LINUX COMMANDS

ls: list all the files and directory
Clear: clear the terminal screen.
pwd: Display working directory path.

mydir: Create a new directory
mydir subha

Directory → Directory is a Mechanism which stores files.

cd: Enter inside the directory

cd ..: Exist from a directory.

man:

cp: copy a file.

mv: Rename a file or directory

locate:

whereis:

rm:

rm dir: delete a directory (only delete empty directory)
rmdir subha

rm -rf: delete any non-empty, empty directory.

rm *.c: delete only .c file

rm *.out: delete only out file

rm *: all files will be deleted

rm * -rf: all files and directory will be deleted.

gcc:

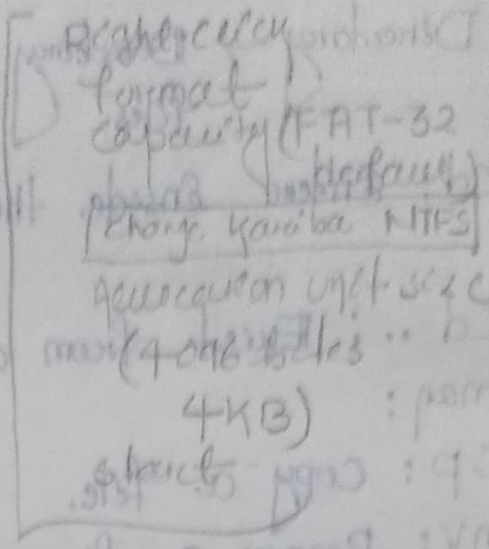
gedit:

valgrind:

su:

ldd:

Formatting :- Changing the file system.



Q what is FAT32 (File Allocation Table 32)

→ The 32-bit version of the FAT file system

→ Employed on windows PCs prior to the more advanced NTFS file system, the FAT32 format is widely used for USB drives, flash memory cards and external hard drives for compatibility between all platforms.

Q what is filesystem

File system or files system is a method and data structure that the operating system uses to control how data is stored and retrieved.

* Every group of data is called file.
* The structure and logic rules used to manage the groups of data and their names is called a file system".

CHAPTER-1

20/09/2022

① What is operators ? what are the different types of operators in c.

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

There are following types of operators to perform diff types of operations in c language.

- Arithmatic operators
- Relational operators
- shift operators
- Logical operators
- Bitwise operators
- Conditional operators
- Assignment operators
- Misc operator

② which operator in c cannot be overloaded in c++.

→ For an example, the sizeof operator returns the size of the object or datatype as an operand. This is evaluated by the compiler. It cannot be evaluated during run time. So we cannot overload it.

→ Also . ;: (dot) ?: (conditional) cannot be overloaded in c++

③ What are the difference between precedence and Associativity.

Precedence	Associativity
precedence decide which operators will be perform 1st in an expression	Associativity decides order of evaluation (same there it is evaluated from Left to Right or Right to Left)

* C language supports 45 no° of operators
* keyboard contains 32 no° of symbols.

All the operators are symbols but all the symbols are not operators

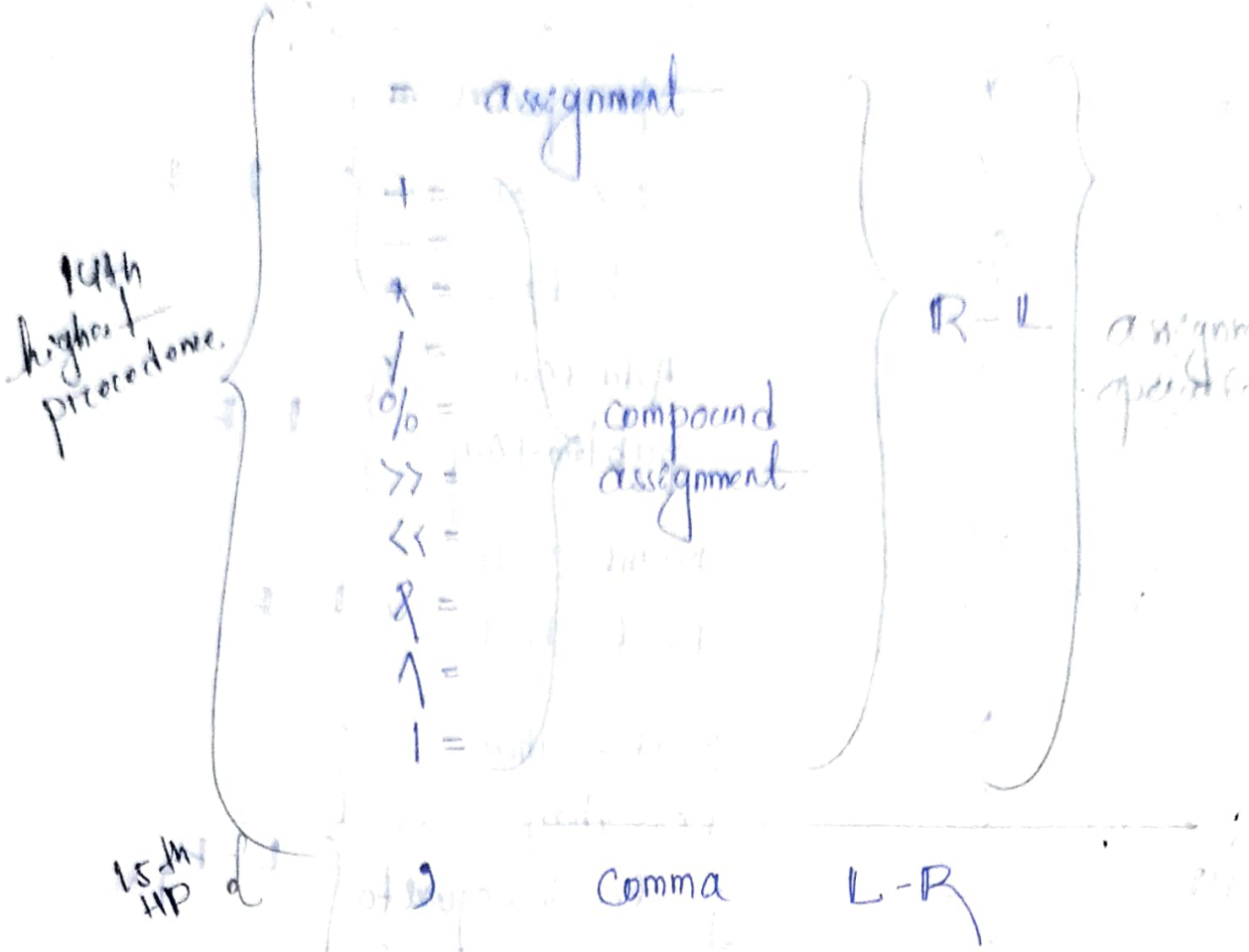
```
int k= 9 * 2 + 7
       |   |
       operand
```

operators is a symbol which operates operand.

```
int k= 9 * 2 + 7
       |   |
       operator
```

	Operators	Description	Associativity
1st highest Precedence.	( )	Function expression	L-R
	[ ]	Array expression	L-R
	.	Structure operator	L-R
	→	"	L-R
2nd highest precedence	++	Increment	R-L
	--	decrement	"
	&	address of	"
	*	in direction	"
	!	not / negation	"
	~	1's complement	"
	size of	Size in bytes	"
	(type)	Type cast	"

	Operator	Description	Associativity	
3rd Highest precedence.	*	Multiplication	L-R	Arithmetic operators
	/	Division		
	%	Modulus		
4th HP	+	Addition	L-R	
	-	Subtraction		
5th HP	>>	right shift	L-R	
	<<	Left shift		
6th HP	>	greater than	L-R	Relational operators
	<	less than		
	>=	greater than equal to		
	<=	less than equal to		
7th HP	!=	not equal to	L-R	
	==	equal to		
8th HP	&	bitwise AND	L-R	bitwise operators
9th HP	^	bitwise XOR	L-R	
10th HP	|	" OR	L-R	
11th HP	||	logical OR	L-R	logical operators
12th	&&	" AND	L-R	
13th	?:	Conditional	R-L	conditional operator.



Relational operator

- >, <, >=, <=, !=, ==
- Return value = 0 or 1.

Program :-

① #include "stdio.h"
main()
{
int k = 90 == 90;
printf("%d", k);
}

14th 7th
k = 90 == 90
90 == 90 (T or 1)
k = 1 (output)

② #include "stdio.h"
main()
{
int a = 5, b = 5, c = 5;

```
if (a==b==c)
printf (" Delhi chalo");
else
  printf (" Ghara. chalo");
}
```

Ghara. chalo
a=, b=, c
(false or 0)

```
if (a == b)
printf (" Delhi chalo");
else
  printf (" Ghara chalo");
```

Delhi chalo
a b
(True or 1)

③
```
#include "stdio.h"
main( )
{
  int k=1==6>3;
  printf("%d", k);
}
```

1h 6
k = 1 (==) 6 (>) 3
1 == 3 (T or 1)
Output
k=1

④
```
#include "stdio.h"
main( )
{
int i=1;
while (i<=5)
{
printf (" Hello mam \n");
i++;
}
```

1 <= 5
Hello mam
2 <= 5
Hello mam
3 <= 5
Hello mam
4 <= 5
Hello mam
5 <= 5
Hello mam
6 <= 5 X

6 times while value will be executed but hello mam will be displayed 5 times.

Logical Operators

(||) (&&)

- ||, &&, !(and)
- Return value 1 or 0.

> NOTE:
> In C language non zero value represent true but zero represent false.

①
```
#include "stdio.h"
main()
{
int a = 90 || 20;
int b = 90 || 0;
int c = 0 || 20;
int d = 0 || 0;
printf("%d %d %d %d", a, b, c, d);
}
```

A quantity which does not equal to zero is said to be non zero.

output
1 1 1 0

```
int e = 90 && 20;
int f = 90 && 0;
int g = 0 && 20;
int h = 0 && 0;
printf("%d %d %d %d", e, f, g, h);
```

output
1 0 0 0

②
```
#include "stdio.h"
main()
{
int i = 5;
int k = 90 || i++
printf("%d %d", k, i);
}
```

output
1 5

In case of OR operator if left side true right side not executed

```
#include "stdio.h"
main()
{
int i=5;
int k=0 || i++;
printf("%d %d", k, i);
}
```

output

1 6

Left side false then the right side will be executed.

(3)
```
#include "stdio.h"
main()
{
int k, m=5;
k= 90 || m=100.
printf("%d %d", k, m);
}
```

k = 90 || m = 100
1

k = (1 = 100)

O/P = error occurs because left side must have any variable.

```
#include "stdio.h"
main()
{
int k, m=5;
k= 90 || (m=100);
printf("%d %d", k, m);
}
```

output

k m

O/P= 1 5

As the left side is true right side is not executed.

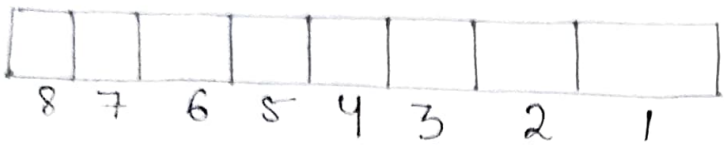
```
int k, m=5;
k= 0 || (m=100);
```

k m

O/P=1 100

left false so right executed

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001
10 = 1010
11 = 1011
12 = 1100
13 = 1101
14 = 1110
15 = 1111



Bitwise operator

& , ^ , | , >> , <<

Bitwise operator works on binary value of a no.

①
```
#include "studio.h"
main()
{
int a = 12 & 9
int b = 13 ^ 10 → Similar = false
                  dissimilar = true
int c = 8 | 5
printf("%d %d %d", a, b, c);
}
```

output

8 7 13

12 = 1100
9 = 1001
1000 (8)

13 = 1101
10 = 1010
0111 (7)

8 = 1000
5 = 0101
1101 (13)

& = Anyone false is false
^ = Dissimilar are true
| = Anyone true is true

`if a. no: (n & n-1) == 0` (power of 2)

② `#include "stdio.h"`
`main()`
`{`
`int n;`
`printf("enter any number");`
`scanf("%d", &n);`
`if((n & n-1) == 0)`
`printf("power of 2");`
`else`
`printf("Not power of 2");`
`}`

10 (not power of 2)
10 = 1010
9 = 1001
1000

8 (power of 2)
8 = 1000
7 = 0111
0000

21.04.2022

Bitwise operators

(Rightshift = division)

n >> 1 = n/2^1 = n/2
n >> 2 = n/2^2 = n/4
n >> 3 = n/2^3 = n/8

(Leftshift = Multiplication)

n << 1 = n * 2^1
n << 2 = n * 2^2
n << 3 = n * 2^3

① `#include "stdio.h"`
`main()`
`{`
`int k = 20 >> 2 << 5 >> 1;`
`int m = 20/4 * 32/2;`
`printf("%d %d", k, m);`
`}`

k = 20 >> 2 << 5 >> 1
5
160
80

O/P
k = 80
m = 80

②
```
#include "stdio.h"
main()
{
int x= 8/2;
int m= 8>>1;
printf("%od%od", x, m);
}
```

conditional operator

? :

① Check a number is even or odd using condition.

```
#include "stdio.h"
main()
{
int n;
printf("enter a number");
scanf("%od", n);
if(n%2==0)
printf("even");
else
printf("odd");
}
```

O/P

4 = even

7 = odd

② Check a number is even or odd using conditional operator.

```
#include "stdio.h"
main()
{
int n;
printf("enter a number");
scanf("%d", n);
```

```
n%2==0 ? printf("even") : printf("odd");
}
```

O/P

23 - odd

42 - even

③ Check a number is even or odd without using conditional operator or condition.

```
#include "stdio.h"
main()
{
int n;
printf("enter a number");
scanf("%d", &n);
n%2==0 && printf("even") || printf("odd");
}
```

1 even (1) = 1

0

&& = Left true right executed

|| = Left true right not executed

&& - returns true when both the conditions are true otherwise false

|| - returns true even if one and both the conditions are true otherwise false

*
```
#include "stdio.h"
main()
{
k = sqrt(36)
printf("%d", k);
}
```

O/P ⇒ k = 6

*
```
#include "stdio.h"
main()
{
int a = printf("Hello");
printf("%d", a);
}
```

O/P

Hello 5

Compound assignment operation

+=, -=, /=, *=, %=, >>=, <<=, &=, ^=, |=

(R-L)

e = 5

e += 3
e = e + 3
= 5 + 3
e = 8

(1)
```
#include "stdio.h"
main()
{
int e = 5;
e += 3;
printf("%d", e);
}
```

O/P
e = e + 3
= 5 + 3
e = 8

(2)
```
main()
{
int e = 8;
e += e *= e /= 2;
printf("%d", e);
}
```

O/P
e = e/2 = 8/2 = 4
e = e * e
= 4 * 4 = 16
e = e + e
= 16 + 16 = 32

*
```
int a = 5, b = 7;
printf("%d %d \n", a, b);
```

O/P
a = 5
b = 7

Swap :-
```
#include "stdio.h"
main()
{
int a = 5, b = 7;
printf("%d %d", a, b);
```

```
// swap
int c = a;
    a = b;
    b = c;
    printf("%d %d\n", a, b);
}
```

output

5 7

7 5

- ```
int a = 5, b = 7
printf("%d %d\n", a, b);
// swap
a = a + b;
b = a - b;
a = a - b;
printf("%d %d\n", a, b);
```

O/P

a (12) = a + b

b = a - b

b (5) = 12 - 7

a (7) = 12 - 5

- ```
#include "stdio.h"
main()
{
int a = 5, b = 7;
printf("%d %d\n", a, b);
// swap
a = a ^ b;
b = a ^ b;
a = a ^ b;
printf("%d %d\n", a, b);
```

$a = a \wedge b \quad b = b \wedge a \quad a = a \wedge b$

2 = 0010 7 = 0111 5 = 0101

5 = 0101 2 = 0010 7 = 0111

a = 0111 = 7 b = 0101 = 5 a = 0010 = 2

O/P

5 7

7 5

Arithmatic operator :-

+, -, *, /, %, ++, -- (L-R)

```
#include "stdio.h"
main()
{
int n, r;
r = 325 % 10; // 5
n = 325 / 10; // 32
```

```
r = 32 % 10;  // 2
n = 32 / 10;  // 3

r = 3 % 10;   // 3
n = 3 / 10;   // 0
```

10 | 32 | 32(1)
   | 30 |
   (2) %

10 | 3 | (0) (1)
   | 0 |
   (3) %

10 | 32 | 3 (1)
   | 30 |
   (2) %

10 | 3 | (0) (1)
   | 0 |
   (3) %

*
```
#include "stdio.h"
main()
{
int n,r;
r = 12%2;  // 0
n = 12/2;  // 6

r = 6%2;   // 0
n = 6/2;   // 3

r = 3%2;   // 1
n = 3/2;   // 1

r = 1%2;   // 1
n = 1/2;   // 0
```

1100
(12)

*
```
#include "stdio.h"
main()
{
int e = 5;
e++;
printf("%d", e);   // 6
int j = 5;
++j;
printf("%d", j);   // 6
```

```
int k=5;
k--;
printf("%d", k); //4
int l=5;
--l;
printf("%d", l); //4
}
```

O/P

6 6 4 4

```
#include "stdio.h"
main()
{
int i;
for(i=1; i<=5; i++)
{
printf("\thello\n");
}
int j;
for(j=1; j<=5; ++j).
{
printf("hi\n");
}
}
```

O/P

hello
hello
hello
hello
hello

---

hi
hi
hi
hi
hi

```
int i=5;
int k= ++i + ++i + ++i + ++i;
printf("%d %d", i, k);
```

then total i=7

7 → 8 → 9

9+8+7+6

i=9
k=31

14+8+9
=31

que en fedora eige

que en ubuntu 8-

9 + 9 + 9 + 9 = 36

++i + ++i + ++i + ++i

i=9
k=36

Gce En Purboc

$9+8+7+6$

$=30$

$x = 30$

$\xi = 9$

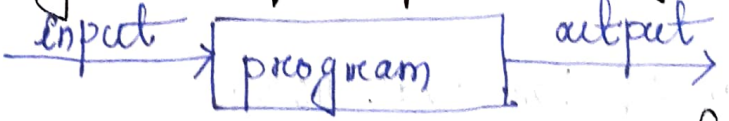
CHAPTER - 2

1. What is datatype abstraction?
2. What is the diff^n betⁿ datatype and modifiers
3. Can a modifier handle all of a datatype
4. What is endianness
5. Diff^n betⁿ variable and constant
6. Which programming language doesn't support datatypes
7. Why we data stores in memory in 2's complement
8. Differentiate betⁿ constant and volatile.

Dt:- 22.04.22

datatypes	qualifiers	modifiers
char	constant	signed
int (%d)	volatile	unsigned (%u)
float (%f)		short
double (%lf)		long (%ld)
void		
enum		
struct		
union		

* Every program takes input and gives output.



* Variable are created in memory not in file.
* All the files are created in disk.
* Storage < memory / File

* Difference between Memory and File.

Memory	File
→ Stores data temporarily	→ stores data permanently
→ When a program is terminated the memory is lost and thus it stores data temporarily	

Format specifiers

int %d unsigned %u char %d
float %f long %l
double %lf long long %ll

// addition of 2 integers :-

```
#include "stdio.h"
main()
{
int num1, num2, res;
printf("Enter 1st no:");
scanf("%od", &num1);
printf("Enter 2nd no:");
scanf("%od", &num2);
res = num1 + num2;
printf("%od", res);
}
```

O/P

Enter 1st number: 123
Enter 2nd number: 10
res 133.

// addition of two real no's :-

```
#include "stdio.h"
main()
{
float num1, num2, res;
printf("Enter the 1st no:");
```

```
scanf ("%f", num1);
printf ("Enter the 2nd no:");
scanf ("%f", num2);
res = num1 + num2;
printf ("%f", res);
}
```

O/P

1.2
10.5
19.7

All scripting programming language does not support data types:-

Python, javascript, html, perl, php, swift.....

// addition of two numbers. (IPYTHON PROGRAM)

```
num1 = input ("Enter 1st no")
num2 = input ("Enter 2nd no")
res = num1 + num2
print res
```

O/P

19
1
20

// calculate area of a circle (C - program)

```
#include "stdio.h"
main()
{
float radius, area;
printf ("Enter radius");
 scanf ("%f", radius);
area = 3.14 * radius * radius;
printf ("%f", area);
}
```

O/P

radius = 8.23
area = 212.681

PYTHON PROGRAM

```
radius = input ("enter radius")
area = 3.14 * radius * radius
print area
```

O/P

radius = 8.23
area = 212.681

Addition of two numbers.

```
#include "stdio.h"
{
int num1, num2, res;
printf("Enter 1st no.");
scanf("%d", &num1);
printf("Enter 2nd no");
scanf("%od", &num2);
res = num1 + num2
printf("%od", res);
}
```

O/P

```
34
23
57
---
340
543
588
---
1234
1235
2469
---
4295060720
       5
---
4295060725
```

no results use other modifiers.

```
#include "stdio.h"
main()
{
long long unsigned int num1, num2, res;
printf("Enter 1st no");
scanf("%llu, &num1);
printf("Enter 2nd no");
scanf("%llu, &num2);
res= num1 + num2
printf("%llu", res);
}
```

Character type

The main objective of the Character type is to process a string and file.

```
#include "stdio.h"
main()
{
char num1, num2, res;
printf("enter 1st no");
num1 = getchar();
getchar();
printf("enter 2nd no");
num2 = getchar();
res = (num1 - 48) - (num2 - 48);
printf("%d", res);
}
```

* Anything consists ~~of~~ in double code is string
* "alok babu likes cocanola"

Reverse yale kola
haba jodé sei line
Re thiga kola then answer
kola haba otherwise 2 dollar

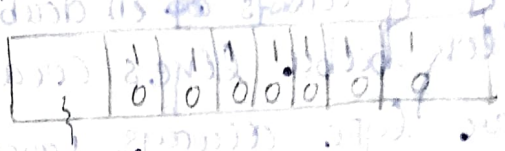
* char type always reserve 1 bytes one 8 bits from the memory.
* There are 2 types of characters:
  (i) Signed.
  (ii) Unsigned.

```
#include "stdio.h"
main()
{
signed char x;
unsigned char y;
printf("%d %d", sizeof(x), sizeof(y));
}
```

Unsigned : 8 bits Signed 8 bits

0 = +ve.

Range of signed +ve (0 - 127)



max = 1111111 = -127 ~~-128~~

min = 0000000 = -0 ~~-1~~

2's complement of 0000001 (-1) = 1111111

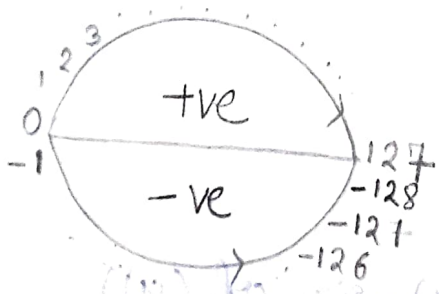
0000000

1111111

+ 1

10000000 -128

2's complement



max = -1 signed -ve

min = -128

Rang -ve signed = -128 to -1

→ To overcome the problem (-0) all the negative data stores in memory in a 2's complement.

→ 27th position is signed bit position.

→ 28th position is signed bit position.

```
#include "stdio.h"
main()
{
unsigned char y=0;
while(1)
{
printf("%d\n",y);
y++;
};
usleep(90000);
}
```

O/P

0-255, 0-255.....

(Control c dare stop hoba)

```
#include "stdio.h"
main()
{
unsigned char i=1;
while(i<=350)
{
printf("Hello");
i++;
}
```

O/P

Entente temes Hello.....

i=0..... 255 then again 0.... 255 so entente temes ef confenues.

```
#include "stdio.h"
main()
{
signed char x=0;
while(1)
{
print("%d\n",x);
x++;
usleep(90000);
}
```

O/P

0

127

-128

-1

0

127

```
#include "stdio.h"
main()
{
while(1)
{
printf("%d\n",x);
if(x&128)
printf("1\n");
else
printf("0\n");
x++;
usleep(90000);
}
}
```

* Unsigned modifier double the range of a datatype

25/4/2022

Float Types :-

- Float type of variable is capable of holding a number which has integer part and decimal part.

Float = 4 byte	= 32 bits
double = 8 byte	= 64 bits

- Double is nothing but long float.

float < double < long float

①
```
#include "stdio.h"
main()
{
float r,a;
printf("enter radius");
scanf("%f",&r);
a = 3.14 * r * r;
printf("%f",a);
}
```

O/P
enter radius
31
a = 3,017.54

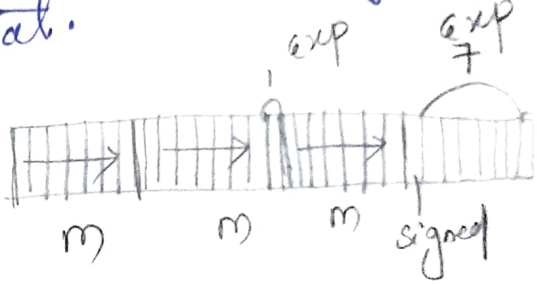
(2)
```
#include<stdio.h>
main()
{
float x = 3.7;
if(x==3.7)
    printf("Hi");
else
    printf("Bye");
}
```

O/P
Bye.

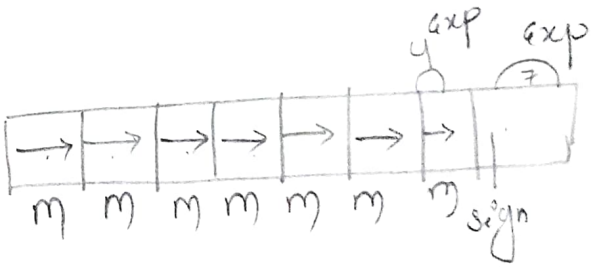
```
if(x==3.5)
if(x==3.5)
```
O/P Hi

Float Memory Representation

Every float or double type stores in memory in 'mantisa' and 'exponet' format.



- float 32 bits
  - 1 bit signed
  - 8 bits exponet
  - 23 bits Mantisa
- double 64 bits
  - 1 bit signed
  - 11 bits exponet
  - 52 bits mantisa



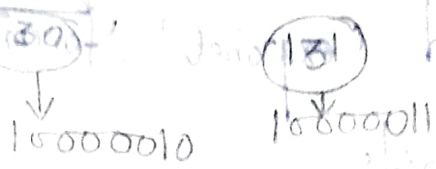
* Every data stores in memory in a binary format.

(3)

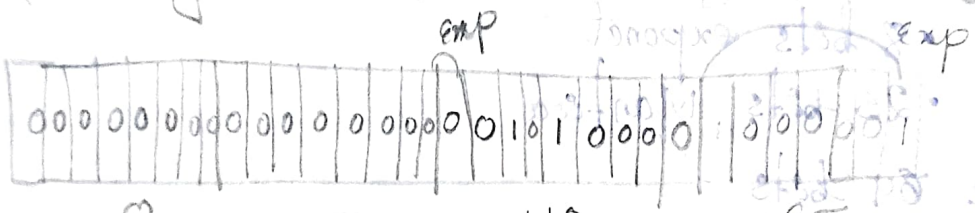
```
main()
{
float x=10.5;
}
```

[NOTE]

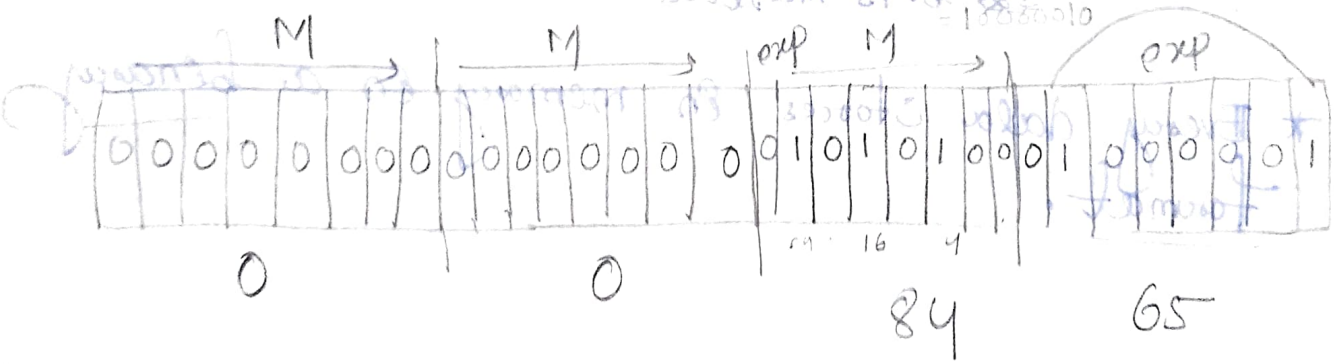
In case of float type 127 is added with exponet, but in case of double type 1023 is added with exponet value



* Remaining bit of mantisa field is 0.



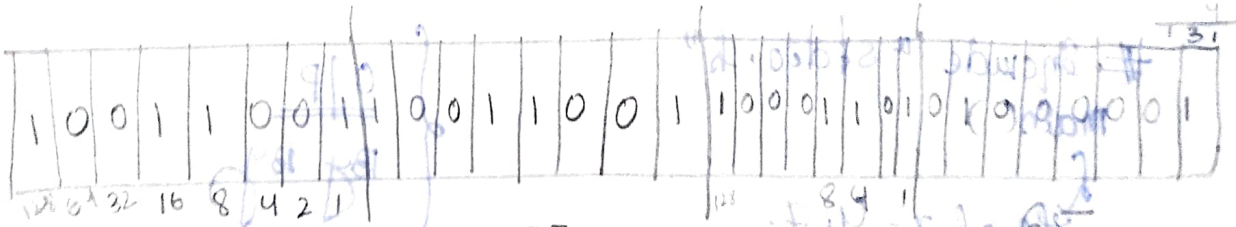
→ x = 13.25 13 = 1101.01 = $1.\underbrace{10101}_{M} \times e^{\overbrace{3}^{130}}$



[NOTE]

* In case of non-recurring float other bits of mantisa filled with zeros.
* In case of recurring float other bits of mantisa filled with recurring value.

(4)
```
#include "stdio.h"
main()
{
float x=17.7;
unsigned char *p=&x;
printf("%d", *(p+0));
printf("%d", *(p+1));
printf("%d", *(p+2));
printf("%d", *(p+3));
}
```



153 153 141

2-12.9

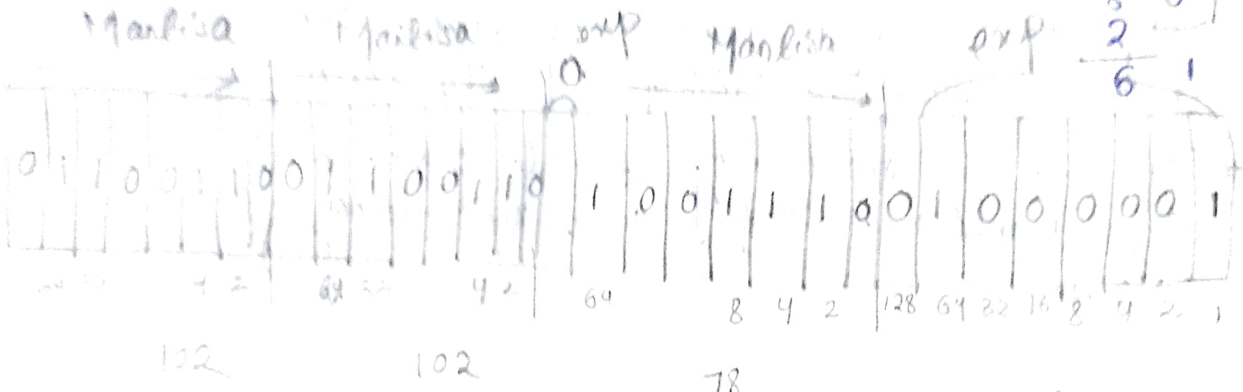
12- 1100.11100

= 1.10011100 $\times e^3$

127
3
130
10000010 exp

.9
2
.8 1
2
6 1
2
2 1
2
0.4 0
2
8 0
2
6 1

Remainder



102 102 78 65

(5)

```
#include "stdio.h"
main()
{
 float x = 4.7;
 if (x = 4.7f)
  printf ("Hi Hi");
else
  printf ("Bye Bye");
  }
```

O/P

Hi Hi

```
#include "stdio.h"
 main()
  {
   float x = 4.7;
   if (x == 4.7)
     printf ("Hi Hi");
else
  printf ("By By");
  }
```

O/P

By By

(recording so we have to change)

(6)
```
#include "stdio.h"
main()
{
float x = 4.5;
if (x == 4.5f)
printf ("hi hi");
else
printf ("Bye Bye");
}
```

O/P

hi hi (bcz of non-recursive)

(7)
```
#include "stdio.h"
main()
{
float x = 4.5;
if (x == 4.5)
printf ("hi hi");
else
printf ("By By");
}
```

O/P

hi hi (bcz of non-recursive)

If the background details of a datatype is hidden from users called datatype abstraction.

float = %f or %g

(8)
```
#include "stdio.h"
main()
{
float x = 9.8;
printf ("%d", x);
}
```

O/P

Garbage value.
%d doesnot know float value.

(9)
```
#include "stdio.h"
main()
{
int x = 9;
printf("%f", x);
}
```

O/P

Garbage

(10)
```
#include "stdio.h"
main()
{
float x = 9.8;
printf("%f", x);
}
```

O/P

9.800000 (%f)

(11)
```
#include "stdio.h"
main()
{
float x = 9.8;
printf("%g", x);
}
```

O/P

9.8 (%g)

26.04.2022

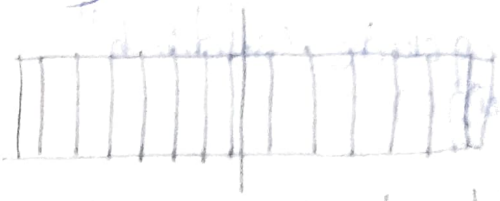
Integer Type

Integer type is 4 types

(a) short int (2 byte, 16 bits)
(b) long int (4, 32 bits)
(c) long long int (8, 64 bits)
(d) int (4, 32 bits)

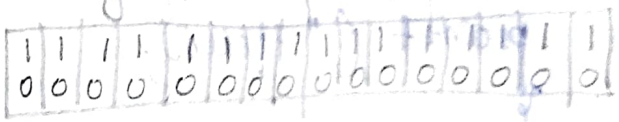
```
#include "stdio.h"
main()
{
short int x;
}
```

O/P (4 byte)



signed data can store total 16 bits.

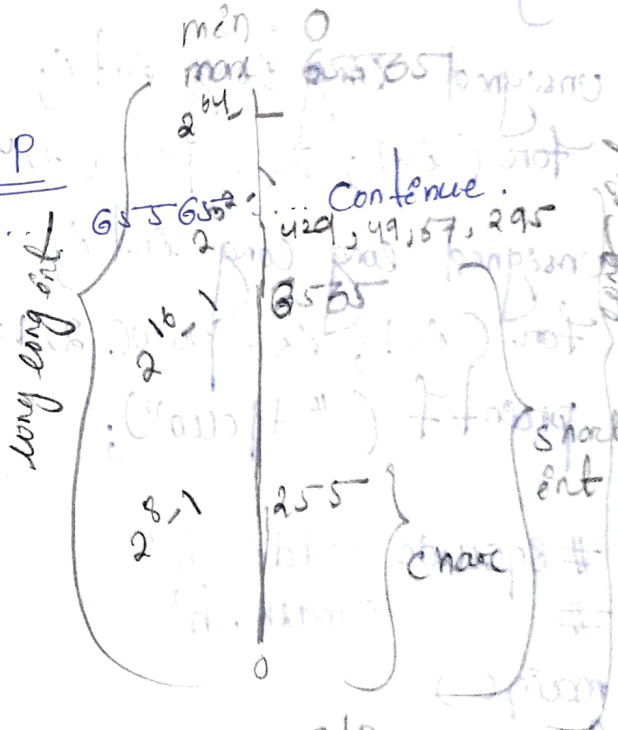
unsigned short int x;




```
#include "stdio.h"
main()
{
unsigned short int x=0;
while(1)
{
printf("%d\n", x);
x++;
usleep(400);
}
```

O/P

0 . . . 65535




```
#include "stdio.h"
main()
{
unsigned short int i;
for (i=1; i<=50000; i++)
printf("Hello");
}
```

O/P

Hello 50000 times
then Out of range
be stop

```
#include "stdio.h"
main()
{
unsigned long int i;
for (i=1; i<=70000; i++)
printf ("Hello");
}
```

O/P

Infinite times bcz, 70000 is greater than 65536

```
#include "stdio.h"
main()
{
unsigned long int i;
for (i=1; i<=pow (2,40); i++)
printf ("Hello");
}
```

O/P

infinite times because $2^{40}-1$ Raw value 358 thu bada

```
unsigned long long int i;
for (i=1; i<=pow (2,40); i++)
```

```
unsigned long long int i;
for (i=1; i<=pow(2, 100); i++)
printf ("Hello");
```

$2^{64}-1$ maximum au adhega nahi

eg:- $2^{10}=2^5 \times 2^5$
$=2^6 \times 2^4$

$2^{100}=2^{50} \times 2^{50}$

```
#include "stdio.h"
#include "math.h"
main()
{
unsigned char i, n1 = pow(2,5);
unsigned char j, n2 = pow(2,5);
for (i=1; i<=n1; i++)
{ for (j=1; j<=n2; j++)
```

O/P

2^{10} times

```
{
printf ("Hello");
}
}
```

```
#include "stdio.h"
#include "math.h"
main()
{
unsigned long long i, n1 = pow(2,50);
unsigned long long j, n2 = pow(2,40)
for (i=1; i<=n1; i++)
{
for (j=1; j<=n2; j++)
{
printf ("Hello");
}
}
```

O/P

2 do Qomes.

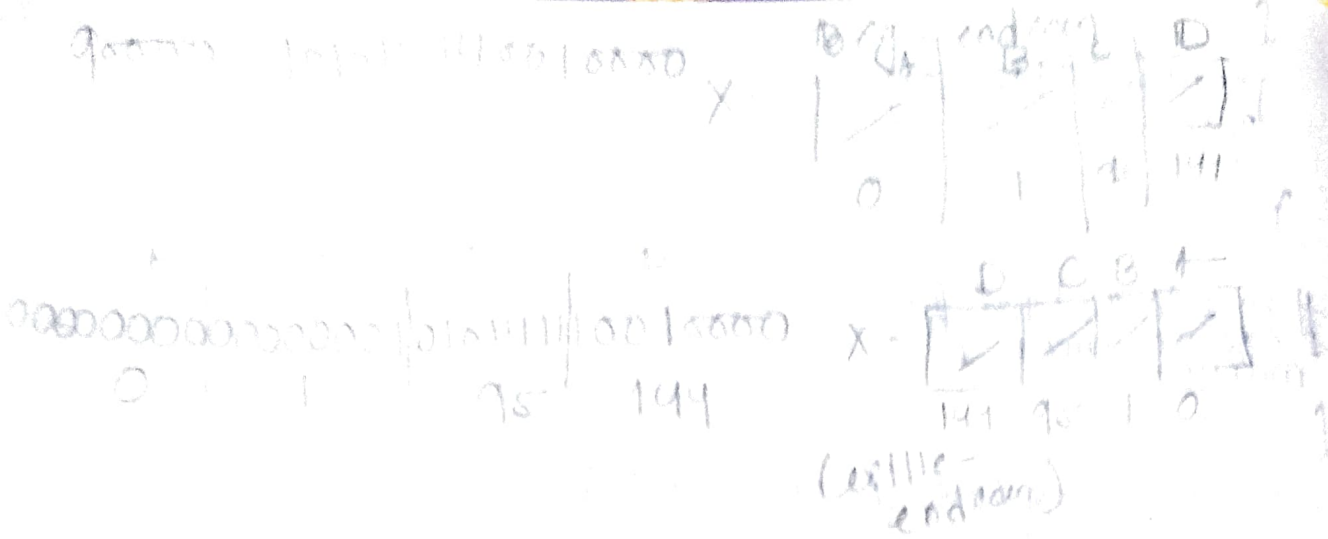
```
main()
{
unsigned long long int i, n1 = pow(2,90);
for (i= 1; i<=n1; i++)
{
printf ("Hello");
}
}
```

O/P

infinite.

Memory Representation of integer

```
#include "stdio.h"
main()
{
unsigned int x = 90000;
}
```



- endean is a process of byte ordering.
- All intel processors are little endean processor.
- Motrola processors are big-endean processor.

```
#include "stdio.h"
main()
{
unsigned int x = 90000;
unsigned char *p = &x;
printf ("%d", *(p+0));
printf ("%d", *(p+1));
printf ("%d", *(p+2));
printf ("%d", *(p+3));
}
```

O/P

141 95 1 0

(intel compiler)

Difference between data types and modifiers.

Data type decides the type of the variable but modifier decides size and range of the variable.

⇒ Enum data type creates a squence set of integer constants.

```
#include "stdio.h"
main()
{
enum xxx{a, b, c, d, e};
printf("%d %d %d %d %d\n", a, b, c, d, e); // 0 1 2 3 4
enum yy{p=8, q, r, s, t};
printf("%d %d %d %d %d", p, q, r, s, t); // 8 9 10 11 12
}
```

Difference between variable and constant :-

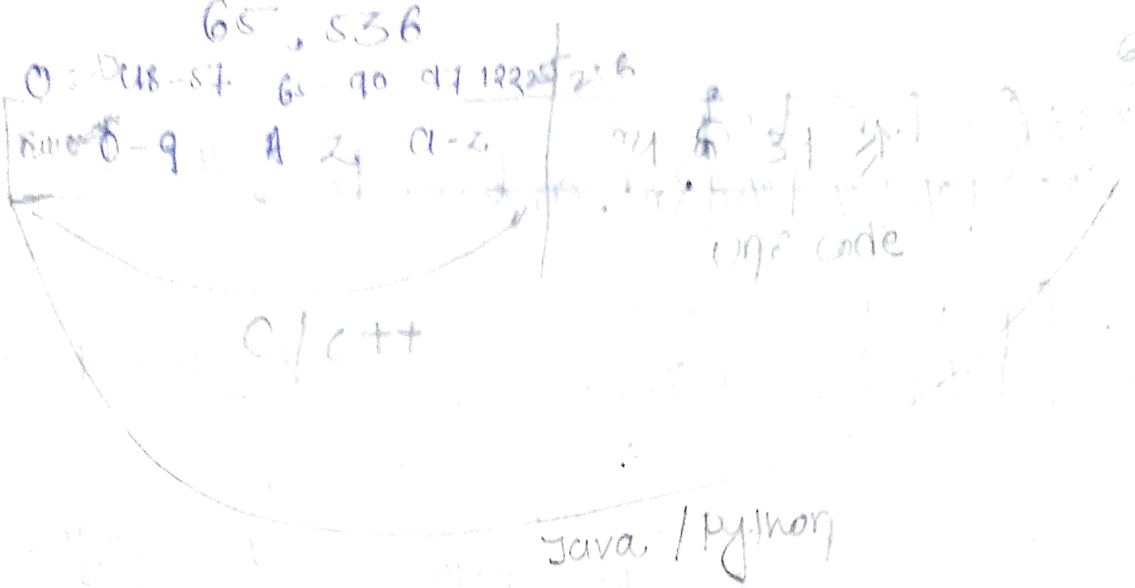
Variable is an object whose value can be change in a program. but constant is a value which cannot be change in a program.

```
x++;
y=y+5;
z++;
m=m+1;
```
✓ variable

```
90++;
4.5f=4.5f+3;
```
✗ constant

```
main()
{
int x=90;
float y=4.5f;
double z=4.5;
char m='A';
```

Character set : ASCII character set



Escape character

\a - 7
\b - 8
\t - 9
\n - 10
\v - 11
\f - 12
\r = 13

Range (7 - 13)

```
#include "stdio.h"
main()
{
printf("%d", 'a');
}
```

O/P
97

```
printf("%d", '\a');
```

O/P
7

CHAPTER-3 CONTROL STRUCTURE

(1) how many statement falls under default scope of a control structure.
(2) how to check a number is even or odd without using any control structure.
(3) Difference bet^n for loop and while loop.
(4) " " while loop & do while loop.
(5) " " break and continue statement.
(6) " " break and return statement
(7) " " return and exit() statement.
(8) " " if else and switch case statement.
(9) what is "hanging if" statement
(10) which loop is called an exist-control loop.
(11) what is the use of nested loop.

List of control structure

(a) loop control structure / repeatition control structure
for, while, do while.
(b) selection control structure
switch, case, default.
(c) decision control structure
if, else
(d) jump control structure.
break, return, continue, goto.

List of control structure :- 27/04/2022

→ Curly braces is used to create a scope.

- A single statement comes under a default scope of a control structure.
- To put multiple statements under the scope we curly braces ({ }).

```
main()
{
  int i=1;
  if(i<=5)
  {
    printf("Hi");
    printf("Bye");
  }
  else
    printf("ok");
    printf("goodbye");
}
```

O/P
Hi
Bye
goodbye.

If-else (Decession control structure)

1. WAP to check a number is even or odd.
2. " " calculate age of a person.
3. " " " result as follows.

avg	result
>=60	first
>=50	second
<50	fail

avg = (P1+P2+P3+P4)/4

④ WAP to find charges as follows.

Item	amount
Regular Over time (OT)	8h 500
	next 4h 200ph
	next 4h 300ph

⑤ WAP to find net and gross salary as follows.

basic	ta	da	Pf	gross	net
>=20,000	2%	3%	12%	?	?
>=10,000	0	2%	12%	?	?
<10000	0	0	12%	?	?

gross = basic + ta + da
net = gross - Pf.

```
#include "stdio.h"
main()
{
int i=1;
while(i<=5)
{
printf("Hi");
printf("Bye");
printf("ok");
i++;
}
}
```

O/P

Hi
Bye
ok (5 times)

```
#include "stdio.h"
main()
{
int i=1;
while(i<=5)
{
```

```
printf("Hi");
printf("Bye");
printf("ok");
}
i++;
}
```

O/P

∞ times

Hi
Bye
ok

- ```
#include "stdio.h"
main()
{
int i=1;
while(i<=5)
printf("Hi");
printf("Bye");
printf("ok");
i++;
}
```

O/P

Hi (∞ times)
Bye
ok

How many statement comes under default?
1st statement

- ```
#include "stdio.h"
main()
{
int i=1;
if (i<=5)
{
printf("Hi");
printf("Bye");
{
else
{
printf("ok");
printf("goodbye");
}
}
```

O/P

Hi
Bye.

```
#include "stdio.h"
main()
{
int i = 1;
if (i<=5)
{
printf ("Hi");
printf ("Bye");
}
else
printf ("ok");
printf ("good bye");
}
```

O/P

Hi
Bye
good bye.

28104 2022

```
#include "stdio.h"
main()
{
short int cd, cm, cy;
short int bd, bm, by;
short int td, tm, ty;
printf ("Enter current date");
scanf ("%hd %hd %hd", &cd, &cm, &cy);
printf ("Enter birth date");
scanf ("%hd %hd %hd, &bd, &bm, &by);
// logic goes here
if (cd >= bd)
td = cd - bd;
else
if (cm == 1 || cm == 3 || cm == 5 || cm == 7 || cm == 8 ||
    cm == 10 || cm == 12)
```

```
{
cd = cd + 31;
cm = cm - 1;
bd = cd - bd;
}
else
if (cm == 4 || cm == 6 || cm == 9 || cm == 11)
{
cd = cd + 30;
cm = cm - 1;
bd = cd - bd;
}
else
if (cm == 2)
{
if (cy % 4 == 0)
cd = cd + 29;
else
cd = cd + 28;
cm = cm - 1;
bd = cd - bd;
}
if (cm >= bm)
    bm = cm - bm;
else
{
cm = cm + 12;
cy = cy - 1;
bm = cm - bm;
}
by = cy - by;
```

```
printf("your age now %hd %hd %hd", d, m, y);
}
```

29/04/2022

```
#include "stdio.h"
main()
{
float basic, ta, da, pf, gross, net;
printf("enter basic salary");
scanf("%f", &basic);
if (basic >= 25000)
{
ta = 0.02 * basic;
da = 0.03 * basic;
pf = 0.12 * basic;
}
else
if (basic >= 15000)
{
ta = 0;
da = 0.02 * basic;
pf = 0.12 * basic;
}
else
{
ta = 0;
da = 0;
pf = 0.12 * basic;
}
gross = basic + ta + da;
net = gross - pf;
printf("gross is %f\n", gross);
printf("net is %f\n", net);
}
```

O/P

enter the basic salary = 25000
gross is =
net is =

```
#include "stdio.h"
main()
{
int P1, P2, P3, P4;
float avg;
printf("enter marks P1, P2, P3, P4");
scanf("%d %d %d %d", &P1, &P2, &P3, &P4);
avg = (P1+P2+P3+P4)/4.0;
if (avg > 60)
printf("first");
else
if (avg >= 50)
printf("second");
else
printf("fail");
}
```

```
#include "stdio.h"
main()
{
int time, amount;
printf("enter total time");
scanf("%d", &time);
if (time < 8 || time > 16)
{
printf("Invalid time");
return 0;
}
```

Time	amount
8 h	500
4 h	200 ph
4 h	300 ph

16 h − 8 h = OT (over time)

```
if (time == 8)
amount=500;
else if (time>8 && time<=12)
amount=500+(time-8)*200;
else
if (time>12 && time<=16)
amount=500+800+(time-12)*300;
printf("Total amount is %d", amount);
}
```

Loop control structure

→ Loop control structure is also known as repetation control structure.

For: It is suitable if number of repetation is known.

While: It is suitable if number of repetation is unknown.

dowhile: It is suitable for menu driven program

eg

```
#include<stdio.h>
main()
{
int i;
for(i=1;i<=5;i++)
{
printf("Hello");
}
```

O/P

Hello
5 times.

```
#include "stdio.h"
main()
{
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
}
```

O/P

Hello
Hello
Hello
Hello
Hello

```
#include "stdio.h"
main()
int i;
for(i=1; i<=5; i++)
{
printf("%d\n", i);
}
for(i=1; i<=5; i--)
{
printf("%d\n", i);
}
```

1 2 3 4 5

5 4 3 2 1

Q Display all the odd numbers from 1 to 100

Q " Summation of all even numbers from 1 to 10

Q Write a program to display all the unique numbers after permutation and combination of 3 digits such as 1, 2, 3.

Q all the digits from 0 - 9. (10 nested loop)

Answer :-

```
#include "stdio.h"
main()
{
int i;
for(i=1; i<=100; i++)
{
if(i%2==1)
printf("%d\n", i);
}
}
```

odd no

```
#include "stdio.h"
main()
{
int i, sum=0;
for(i=1; i<=10; i++)
{
if(i%2==0)
sum=sum+i;
}
printf("%d", sum);
}
```

O/P
30

```
#include "stdio.h"
main()
{
int i, j, k;
for(i=1; i<=3; i++)
{
for(j=1; j<=3; j++)
{
for(k=1; k<=3; k++)
{
if(i!=j && i!=k && j!=k)
printf("%d%d%d\n", i, j, k);
}
}
}
```

O/P

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

all even

30/04/2022

1. WAP to find sum of digits of a number.
2. WAP to find sum of smallest and largest digit of a number.
3. WAP to find reverse of a number.
4. To check a number is Armstrong or not.

```
#include "stdio.h"
main()
{
int n, rc, sum = 0
printf("enter any number");
scanf("%d", &n);
while (n > 0)
{
 rc = n % 10;
 sum = sum + rc;
 n = n/10;
}
printf("%d", sum);
}
```

O/P

35

9

```
if (rc % 2 == 0) (find sum of
all the even
digit of a no.)
```

O/P = 4 + 6

```
#include "stdio.h"
main()
{
int n, rc, l=0, s;
printf("enter any number");
scanf("%d", &n);
s = n;
while (n > 0)
{
rc = n % 10;
if (s > rc)
    s = rc;
if (l < rc)
    l = rc;
n = n/10;
```

l = 0

s = n

1972 > 2

s = 2

0 < 2

l = 2

n = 197

R = 7

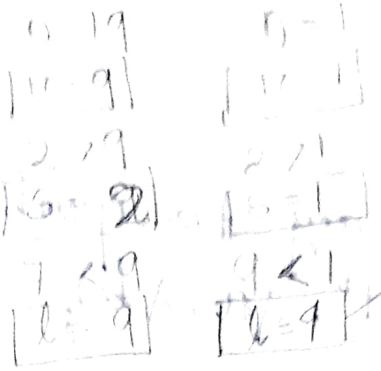
2 > 7 : l = i

s = 2

2 < 7

l = 7

```
}
printf("%d", l+s);
}
```



```
#include "stdio.h"
main()
{
int n, rc, s=0;
printf("enter any number");
scanf("%od", &n);
while (n>0)
{
rc= n%10;
s= s*10 +rc;
n= n/10;
}
printf("%od", s);
}
```

← Reverse

```
#include "stdio.h"
main()
{
int n, rc, s=0, temp;
printf("enter any number");
scanf("%d", &n);
temp= n;
while (n>0)
{
rc= n%10;
s= s*10 +rc;
n= n/10;
}
if (temp==s)
printf("Amstrong");
else
printf("Not amstrong");
}
```

2-08-2022

Q Find out the HCF (Highest common factor) or GCD (greatest common divisor)

```
#include "stdio.h"
main()
{
int a, b;
printf("enter any two numbers");
scanf("%d%d", &a, &b);
while(a!=b)
{
 if(a>b)
    a=a-b;
 else
   b= b-a;
}
printf("%d", a);
}
```

O/P

a-2

a=10, b=12

12 > 10

b = 12 - 10 = 2

a=10, b=2

10 > 2

a = 10 - 2

a = 8

a=8, b=2

8 > 2

8-2=6

a=6, b=2

6-2=4

a=4, b=2

4 > 2

4-2=2

a=2, b=2

```
#include "stdio.h"
#include "math.h"
main()
{
~~decimal number~~
int n, i=1, r, s=0, temp, s1=0;
printf("enter a number");
scanf("%d", &n);
temp = n;
// reverse
while(n>0)
{
r = n%10;
s = s*10+r;
n = n/10;
```

n = 135

135

r = 5

s = 5

n = 13

r = 3

s = 53

n = 1

r = 1

s = 531

n = 0

s = 531

Sum of each digit to the power position is same as given number called a daisercium number.

pow(1,1) + pow(7,2) + pow(5,3)

pow(1,1)

pow(1,1)

```
while (s>0)
{
r = s%10;
s1 = s1 + pow(r, i);
i++;
s = s/10;
}
if (s1 == temp)
    printf ("\n Daisercium number");
else
    printf ("\n Not");
}
```

Dispay odd numbers from 10 to 20

```
{
int i, n;
for (i=10; i<= 20; i++)
{
printf ("%d ", i);
}
}
```

```
#include "stdio.h"
main()
{
int i, n, rc, s;
for(i=10; i<=20; i++)
{
  n=i;
  s=0
// find sum of each digits of n
while(n>0)
{
  rc= n%10;
  s= s+rc;
  n= n/10;
}
printf("%d", s);
}
```

Display all the numbers from 10 to 20 those no whose sum of its digit and the square of no. of digits is same.

```
#include "stdio.h"
#include "math.h"
main()
{
int i, n, rc, s, s1;
for(i=10; i<=20; i++)
{
  n=i;
  s=0;
  // find sum of each digits
  while(n>0)
  { rc= n%10;
    s= s+rc;
    n=n/10;
```

O/P:

10, 11, 13, 20
12

```
n = pow(i, 2);
s1 = 0, sum of cam digits of n
while (n > 0)
{
    r = n % 10;
    s1 = s1 + r;
    n = n / 10;
}
if (pow(s, 2) == s1)
printf("%d", i);
}
}
```

3/05/2022

factorial of a number :-

```
#include "stdio.h"
main()
{
int n, f = 1;
printf("enter a number");
scanf("%d", &n);
while(n > 0)
{
    f = f * n;
    n--;
}
printf("%d", f);
}
```

n = 5
f = 1 * 5
= 5

NOTES

If multiple functionality is implemented in a single program then do-while loop is used

→ do-while loop is suitable for a menu driven program.

// WAP to find gcd, factorial, sum of digit and reverse of numbers.

```
# include "stdio.h"
main()
{
 int n;
 do
 {
  printf("\n1 gcd");
  printf("\n2 factorial");
  printf("\n3 sum digits");
  printf("\n4 reverse");
  printf("\n0 to exit");
  printf("\n enter your choice");
  scanf("%d", &n);
  if (n==1)
  {
   int a, b;
   printf("enter any two numbers");
   scanf("%d%d", &a, &b);
   while(a != b)
   {
    if (a>b)
     a=a-b;
```

```
else
  b=b-a;
}
printf("%d",a);
}
else
if(n==2)
{
int n,f=1;
printf("enter any number");
scanf("%d",&n);
while(n>0)
{
f=f*n;
n--;
}
printf("%d",f);
}
else
if(n==3)
{
int n,rc,sum=0;
printf("enter any number");
scanf("%d",&n);
while(n>0)
{
rc=n%10;
sum=sum+rc;
n=n/10;
}
printf("%d",sum);
}
```

```
if(n
else
if(n==4)
{
int n,rc,s=0;
printf("enter any number");
scanf("%d",&n);
while(n>0)
{
rc=n%10;
s=s*10+rc;
n=n/10;
}
printf("%d",s);
}
while(n!=0);
}
```

Jump control statements

break, continue, return, goto.

```
main()
{
  int i=1;
  while(i<=5)
  {
   printf("Hello");
   break;
  printf("Bye");
  i++;
  }
  printf("Goodbye");
}
```

O/P

Hello
Goodbye.

O/P

Hello
Bye

without breaking

NOTES

break · statement transfer control just outside loop

```
main()
int i=1;
while(i<=5)
{
 printf("Hello");
 return 0;
printf("Bye");
i++;
}
printf("goodbye");
}
```

O/P

Hello.

(Control balaraya jeba hari)

NOTES
Return statement transfer control to end of function

```
main()
{
int i=1;
while(i<=5)
{
printf("Hello");
continue;
printf("Bye");
i++;
}
printf("goodbye");
}
```

O/P
display hello infinite time.

NOTES
Continue statement transfer control again to begening of the loop

- Difference between return and exit statement.

→ Return and exist statement both are same in main function

```
main()
{
int i=1;
while(i<=5)
{
printf("Hello");
exist(0); or return 0;
printf("Bye");
```

```
    i++;
  }
  printf("good bye");
}
```

O/P
Hello.

NOTES

Both both are different in other function but same in the main function.

- exist function terminate the whole program. Return statement terminate the fun.

```
int main()
{
  bbsrc();
  printf("it is main \n");
  cte();
}
int bbsrc()
{
  printf("It is bbsrc \n");
  exit(0);
  printf("it is bbsrc 2 \n");
}
int cte()
{
  printf("it is cte \n");
}
```

Function working

it is bbsrc
it is main
it is cte

exit

It is bbsrc

goto statement transfer the control from one place to another place in a program.

```
#include "stdio.h"
int main()
{
goto x;
y:
printf("FA");
goto z;
x:
printf("LA");
goto y;
z:
printf("IBA");
}
```

O/P

LA FA IBA

Selection control structure

switch, case, default.

```
#include "stdio.h"
main()
{
int x=1;
switch(x)
{
case 1:
   printf("one");
   break;
case 2:
   printf("two");
   break;
case 3:
   printf("three");
   break;
```

O/P (without break)

one
two
three
none

```
default:
    printf ("\nlone");
    }
}
```

NOTES

1. Execution in switch case always starts from match case.
2. If there is no match case, then default case is executed.
3. break statement prevents falling of control from match case to next case.
4. Any problem solve using switch case that can be solve using of else statement.
5. But switch case is faster than if else statement.

Execution in switch case directly starts from match case but execution in if-else always starts from beginning.

// Check a number is even or odd using switch case.

```
#include "stdio.h"
main()
{
int n;
printf ("enter a number");
scanf ("%d", &n);
switch (n%2)
{
case 0:
    printf ("even");
    break;
case 1:
printf ("odd");
}
```

Any program solve in if else can also solve in switch case

```
#include "stdio.h"
main()
int avg = 27;
if (avg > 60)
printf ("first");
else
if (avg > 50)
printf ("second");
else
if (avg >= 40)
printf ("third");
else
printf ("fail");
}
```

If - else

```
#include "stdio.h"
main()
{
int avg = 27;
switch (avg / 10)
{
case 10:
case 9:
case 8:
case 7:
case 6:
printf ("first");
break;
case 5:
printf ("second");
break;
case 4:
printf ("third");
break;
default:
printf ("fail");
}
}
```

Switch case

04-08-2022

```
#include "stdio.h"
main()
{
int n, i, j, k=1, s;
printf("enter the rows");
scanf("%d", &n);
s=n-1;
for(i=1; i<n; i++)
{
// print space
for(j=1; j<=s; j++)
printf(" ");
// print *
for(j=1; j<=k; j++)
printf("*");
// create newline
printf("\n");
s=s-1; (space)
k=k+2; (star)
}
}
```

O/P

enter the rows 6

```
#include "stdio.h"
main()
{
int n, i, s, j;
printf("enter the rows");
scanf("%d", &n);
s=n-1;
```

```
for (i=1; i<=n; i++)
{
  for (j=1; j<=s; j++)
    printf(" ");
  for (j=1; j<=i; j++)
    printf("%d", j);
  for (j=i-1; j>=1; j--)
    printf("%d", j);
  printf("\n");
  s=s-1;
}
}
```

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

05-05-2022

// prime numbers.

```
#include "stdio.h"
main()
{
int n, i, c=0;
printf("enter any number");
scanf("%d", &n);
for (i=10; i<=n; i++)
{
if (n%i==0)
i++;
}
if (c==2)
printf("prime");
else
printf("not prime");
```

```
main()
{
int n, i, c;
for (n=1; n<=100; n++)
{
 c=0;
 for (i=1; i<=n; i++)
 {
 if (n%i==0)
 c++;
 }
 if (c==2)
 printf("%d", n);
 }
}
```

```
#include "stdio.h"
#include "math.h"
main()
{
int n, s=0, rc, e=0;
printf("enter any number");
scanf("%d", &n);
while(n>0)
{
rc= n%10;
s= s+rc * pow(10, e);
e++;
n= n/10;
}
printf("%d", s);
}
```

Q $1/1! + 2/2! + 3/3! + \ldots + n/n!$