Date

```
int main()
{
  int i, j, k
  for (i=1; i<=30; i++)
  {
    for (j=i; j<=30; j++)
    {
      for (k=j; k<=30; k++)
      {
        if (i*i + j*j == k*k || j*j + k*k == i*i || i*i + k*k == j*j)
        {
          printf("\n %d %d and %d", i, j, k);
        }
      }
    }
  }
```

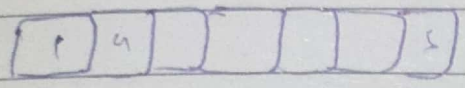
3 4 5
5 12 13

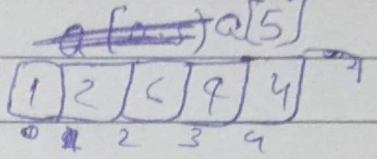
i 1 → 30
j 1 → 30
k 1 → 30

123012

Arrays

Date 3-Nov-21





a[1] = 2

a[0] = 1

An array is a collection of elements of the same type that are referenced by a common name.

Compared to the basic data type (int, float & char) it is an aggregate or drive data type - All elements of an array occupy a set of con type memory location.

a[3] = 4

Date

```
{
  int i;
  int a[5] = {1,2,9,4,5}
  Printf("%d", a[3])
}
```

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

```
while (i<5)
  printf("%d", a[i])
```

```
int i;
int a[5];
for(i=0; i<5; i++)
{ Scanf("%d", &a[i]); }

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

```
#include<stdio.h>
int main()
{ int marks[10], i, n, sum = 0, average

printf("Enter number of element")
scanf("%d", &n);

for(i=0; i<n; ++i)
{
  printf("Enter number %d: ", i+1);
```

```
#include<stdio.h>
int main()
{ int i, n=10
float temp[10]
  for(i=0; i<n; i++)
{ scanf("%f", temp[i]), }
```

1-D Array

int a[5] = {10, 20, 50, 70, 60}

Bit
5 x 4 = 20

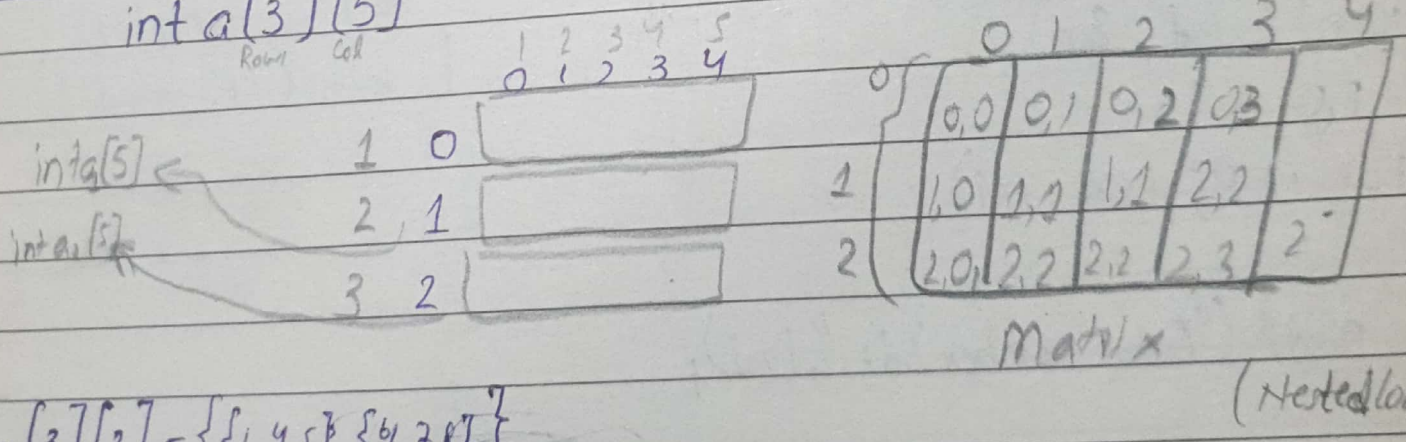
Formula: B.A + (Index x Size)

	0	1	2	3	4	
Loc =	10	20	50	70	60	
	1000	1004	1008	1012	1016	1020

B.A
First Value

2D-Array

int a[3][5]
Row Col



int a[3][2] = {{1,4,5}, {4,7,8}}
a[2][3]

(Nested loop)

{{1,2,3}, {2,5,6}}

	0	1	2
Row 0	1	2	3
Row 1	2	5	6

int a[2][3] = {
{1, 2, 3},
{2, 5, 6}
}

Date

```
int i, j;   Sum=0
int a[2][3];
  for(i=0; i<2; i++)
    {
      for(j=0; j<3; j++)
      {
        scanf("%d", &a[i][j]);  → Sum = Sum + a[i][j]
      }
    }
  for(i=0; i<2; i++)
  {
    for(j=0; j<3; j++)
    {
      printf("%d", a[i][j]);
      Printf("%d%d", a[i][j], sum);
    }
  }
  Return 0;
}
```

Outer loop is always responsible for rows.

CamScanner

Functions

Date 9-Nov

```
main()
{
  message();
  printf("\n Cry, and you stop the monotony");
}
message()
{
  printf("\n Smile, and the world smiles with you")
}
```

Main

Small func

Output

Smile.....

Cry.....

```
main()
{ printf("\n I am in main");
  italy();
  brazil();
  argentina();
}
italy()
{
  printf("\n I am in italy"); }
brazil()
{ printf("\n I am in brazil") }
argentina()
{ printf("\n I am in argentina") }
```

a) C program is a collection of one or more functions.

b) A function gets called when the function name is followed by a semicolon. For example

```
main()
{
  argentina();
}
```

EAGLE

CamScanner

Date

(c) A function is defined when function name is followed by pair of

```
argentina()
{
  statement1;
  statement2;
  statement3;
}
```

d) Any function can be called from any other function. Even main() can be called from other functions. for example:

```
main()
{
  message();
}
message()
{ printf("\n Can't imagine life without C");
  main();
}
```

e) A function can be called any number of times - for example:

```
main()
{   message()
    message()
}
  message()
{ printf("\n Had a good Mid Exam");
```

EAGLE

Date

f)

```
main()
{
  message1();
  message2();
}
  message2();
  {
  printf("\n But the butter was bitter");
  }
   message1()
 {
  printf("\n Mary bought some butter").
```

g)

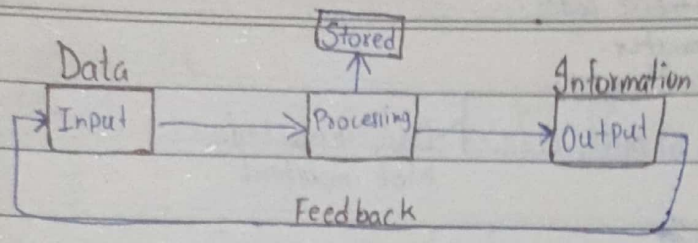
h)

```
main()
{printf("\n I am in main");
 argentina()
   {printf("\n I am in argentina");
   }
}
```

EAGLE

CamScanner

Date 24-Oct-22



:) Data is uncategorized & unsorted
:) Information is categorized and sorted

:) Chunks that are not processed donot have a conclusion is called data.
:) Processed data, that have a conclusion is called information.
:) Feedback is important.

Processing Unit:

2^h binary

ALU: Airthmetic & Logic Unit
CU: Control Unit
MU: Memory Unit (Register)

The bigger the size of register, the faster the processing speed i.e 64bit 128bits etc

The three part of processing unit is connected through buses.

ALU → [+ - / *]

Opcode → Operation that is performed.
Operand → On which opcode is performed on.

ASCII Code:
Binary Convertion

a + b
↓ Operand ↳ Opcode → ALU is used
MU is used

CamScanner

Encoding Schemes:

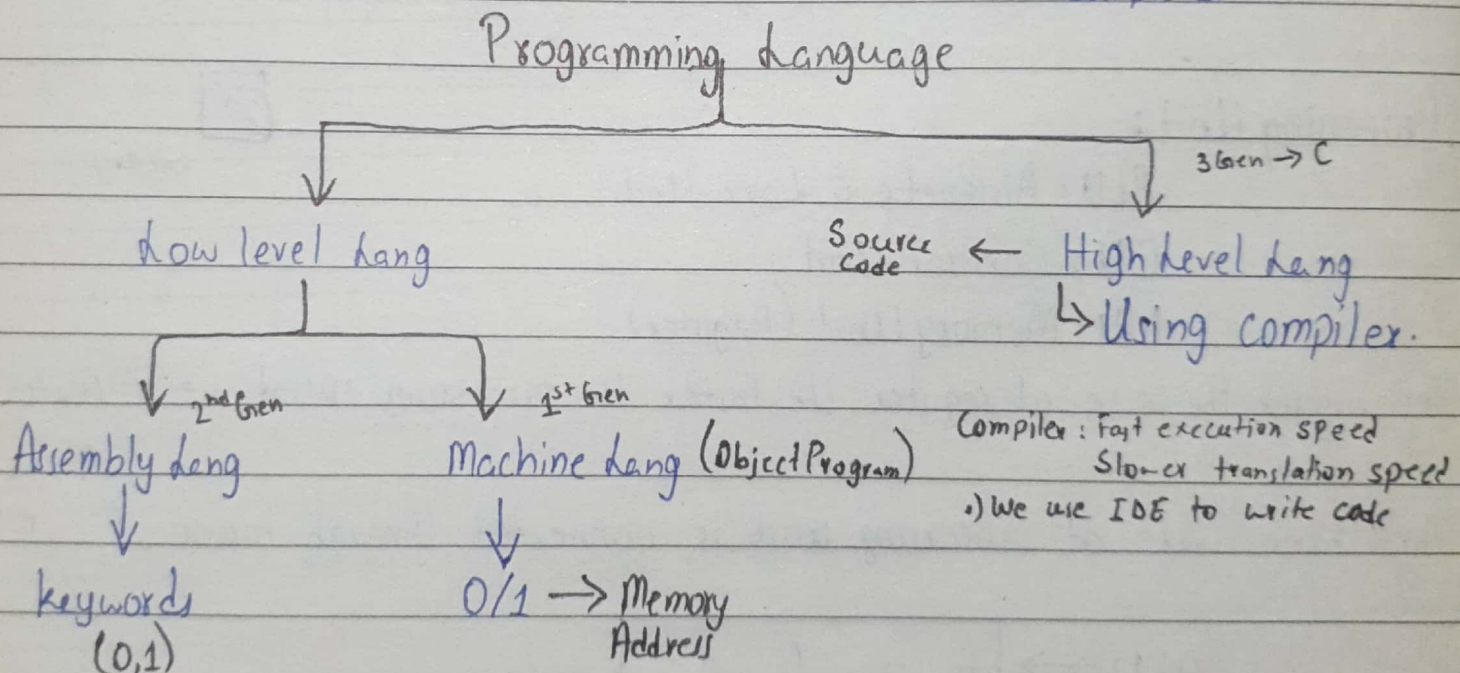
- Numeric Data : Encoded as binary numbers.
- Non-Numeric : Encoded as binary number using representative codes.
  1) ACII - 1 byte per character.
  2) Unicode - 2 bytes per character.

Date

- Logic is basically True/False, Yes or No, 0/1 } Basic knowledge Not important
- Control Unit → Designates the work to the processor and monitors it.
- Airthmatic also uses logical statement.
- A single instruction set contains opcodes & operands - The syntax a single set is different in many programming language

Syntax → Predefined way of writing an instruction set in a programming language

Programming Language → way of communication between human & computer.



- There is no need of any intermediate in machine language.
- Operand is memory address.
- In Assembly language assembler use as a intermidiate.
- LLL is not a user friendly language
- HLL language is a user friendly language.

Date

Components:
•) Hardware •) Software •) lineware •) Firmware

Hardware: CPU, Keyboard, Mouse, Screen, Processor, Tangible device
Storage.

MU consist different devices [MU → MMU → Registers]
↓
Primary Memory, Secondary memory
without primary memory computer can't work but without secondary memory computer can perform.
•) In primary memory we have RAM, ROM, Registers
(MB-GB) (Bytes) (Bits)
• Hard disk is secondary memory but act like primary memory.
• The size of RAM depends on the size of hard disk and power of processor.

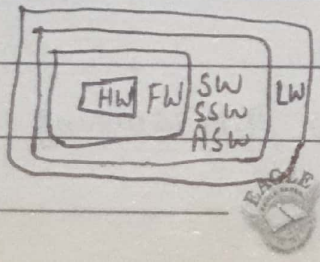
Firmware is a software program or set of instruction programmed on hardware device.
Firmware: is a bootable program which helps to start the computer.
POST
Bootable is in ROM Record is called BIOS (Basic I/O System)
ROM have only address of that bootable record which need for computer to start
Hard Disk have the bootable record.

Liveware: it is a term used to denote people using computers, and is based on the need for a human, or liveware, to operate the system using hardware & software.

Software: is a set of instruction who tells computer what to do and how to do.

There are two types of software.

① System software
Operating System
Windows, linc
Unit



② Application software.
To solve particular problem

Date

For Problem Solving we have:

Flowcharts: (graphically representation)

Algorithms: (in algorithm we create sub programs to solve big problem)

Pseudocodes: (Generic programming language)

Source codes: (Basically a pseudocode but different in every lang)

Write a program which take input from the user & print output.

Start

① Initialize x (reserve memory)

② Take input from user in x

③ Print / output value of x on screen.

End

An PseudoCode:

Start

① Initialize x

② Enter (x) / Input (x)

③ Output (x) / Print (x)

End

Variable / Constant

1B character

2B (int) numeric

4B Float (Decimal number)

Signed integers: Neg & Positive Numbers

Unsigned integers: Positive Number.

•) Float can store integers.

Syntax error = a = b + c +

Operators:

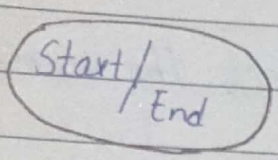
Arithematic operators: +, -, x, ÷

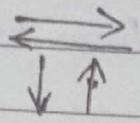
R·H·S = Source L·H·S = Destination

Hard core values don't change in programming cuz it's constant.

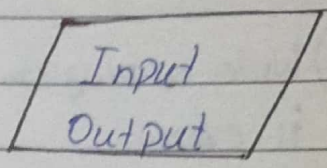
Flow Charts:

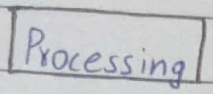
Date _____

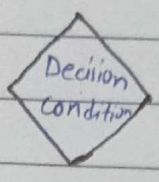


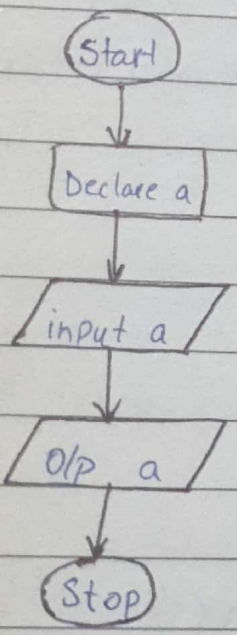


Directional Flow

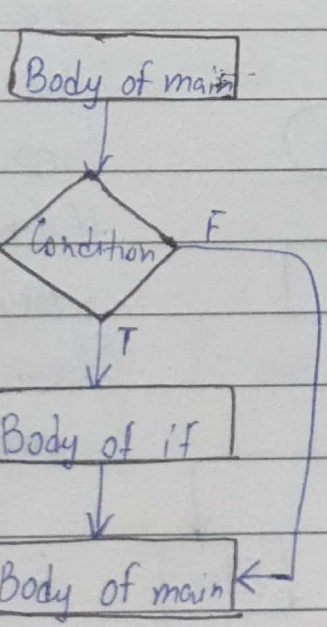








Start
Declare a
Input a
Output a
End.



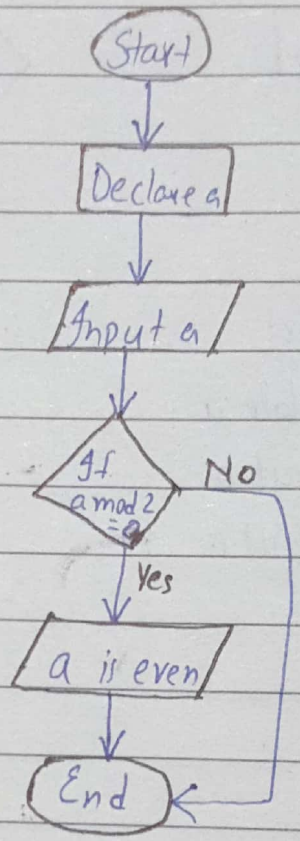

```
if (conditional statement)
{
    Body of if
}
```

Bracket is compulsory if it contain many lin

Date

Q.) Write a program which take input from user & check the number is even.

Start
Declare a
Input a
If a mod 2 = 0
then a is even.
End



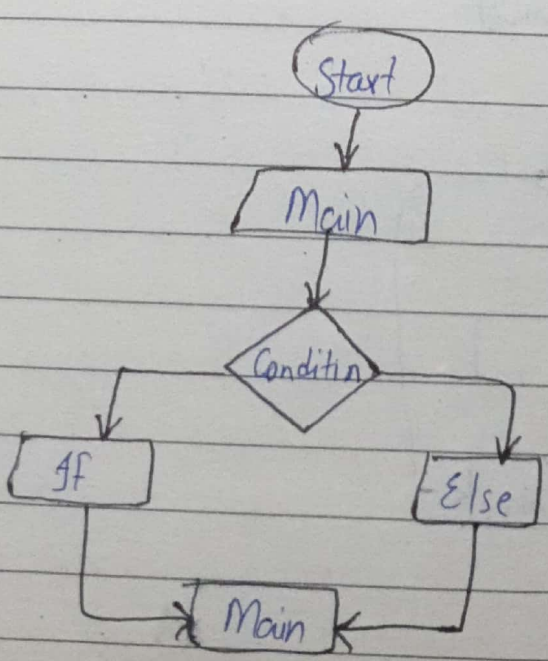

```
{
    int a
    printf("Enter a number");
    Scanf("%d", &a);
    if (a%2 == 0)
    printf("Number is even");
}
```

IF-Else

```
main()
{ // body of main

  if (condition)
  {
  //Body of if
  }
  Else
  { // Body of else
  }

  // body of main
}
```




```
main()
Declare n
input (n)
IF (n%2 == 0)
  printf("num is even");
Else
  printf("num is Odd");
```

Date

```
Start
Declare n
Input (n)
IF (n>0)
   print("num is positive")
Else
   print ("num is negative")
End
```

Q) Write a program which takes sides of triangle as input & check weather the triangle is right angle triangle or not.

```
Declare S1, S2, S3
Print (Input sides of triangle)
Input S1, S2, S3
IF ((S1*S1) == (S2*S2) +
```

Equilateral Triangle:

```
IF (S1==S2 AND S2==S3)
IF (S1==S2 & S2==S3 & S1==S3)
```

```
Declare S1, S2, S3
Print (Input sides of triangle)
Print (S1=b, S2=b, S3=b)
Input (S1, S2, S3)
IF ((S1*S1) == (S2*S2) + (S3*S3))
   print ("It's a right angle triangle")
Else
   print ("Not a right angle triangle").
```

Nested IF

```
IF (Cond: T, F)
IF (Cond: T, IF (cond T, F))
```

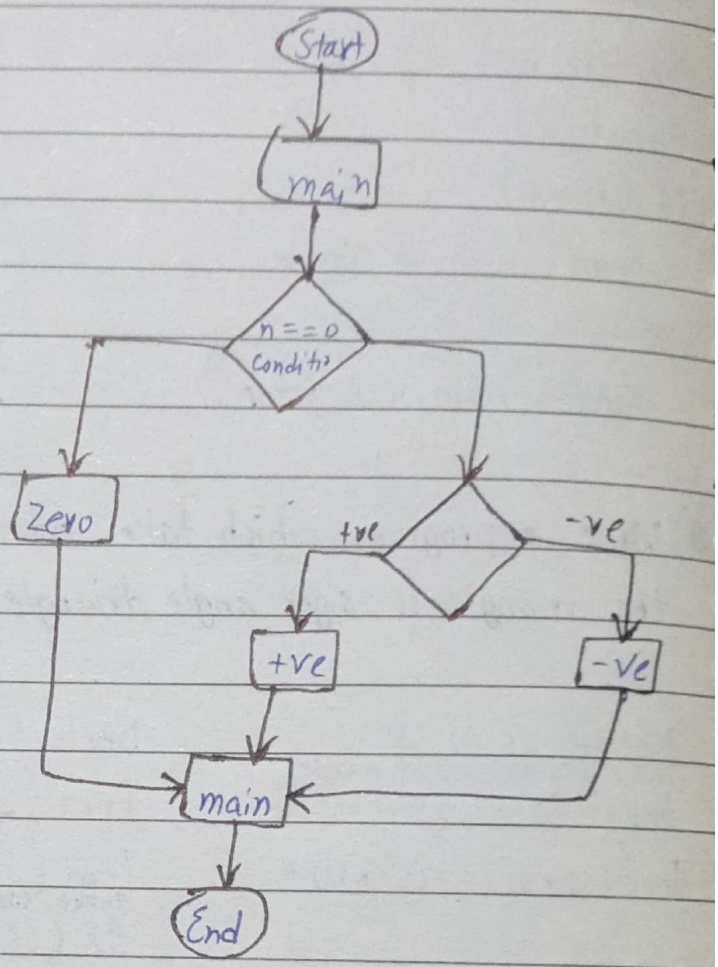
① ② ③

```
IF1 (condition)
   //body of if1
Else
   if2 (condition) //body of else1
      //body of f2
   else2
      //body of else2
   End F2
End F1
```

Date

Num is +ve, -ve or neutral.

```
if (n==0)
{print ("number is neutral")
 }
Else
 { if (n>0)
  {print ("number is +ve")
  }
  else
  {print ("-ve number")
  }
 }
```

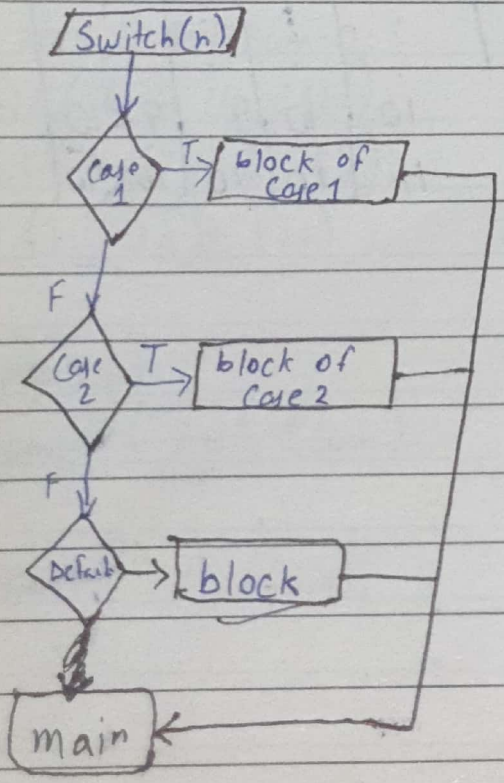



```
Switch (int Button) → int/char → variable
      CASE "ON"
      Case "OFF"
           Case 0
           Case 1
           break;
n=num%2
Switch(n)
{
  Case 0:
      print ("Even")
  break;
  Case 1:
      print ("Odd")
}
```

Enter # 1
Enter # 2
Select operation to perform
Add (A) a
Sub (S) s
Mul (M) m
DIV (D) d
Wrong Input

Date

```
Switch(ch)
{
    Case 'A':
    Case 'a':
        Sum = a+b
        Print(sum)
        break;
Case 'S':
Case 's':
        Sub = a-b
        print(Sub)
        break;
default:
    print(wrong input)
```




```
int
Input num;
Switch (num)
{
Case 1:
    {printf("Monday"); break;}
Case 2:
    {printf("Tuesday"); break;}
Case 3:
    {printf("Wednesday"); break;}
Case 4:
    {printf("Thurday"); break;}
Case 5:
    {printf("Friday"); break;}
Case 6:
    {printf("Saturday"); break;}
Case 7:
    {printf("Sunday"); break;}
default
    {printf("Invalid input");}
}
```

i = 5
x = i++ ; x = 5
x = ++i ; x = 6

Loop: (iterative, conditional) [For] [While (dowhile)]

Date

① Print 2, x number of times, where x is input from user.

```
For (initialization; Condition/iteration checking; inc/dec)
{
}
```

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

i++

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

iteration#	Limit	i (i=i+1)	i<5	Print
1	5	0	0<5	0
2	5	1	1<5	1
3	5	2	2<5	2
4	5	3	3<5	3
5	5	4	4<5	4
6	5	5	5<5	

Global variable: (i)

Jou aik sae zyda use ho.

Local variable: this i is local with respect to for.

↳ Use for certain scope

```
Start
Declare n, x
Input n, x:
for 1 to x:
    print(n)
End for
End.
```

For i = 1 to 10

iteration#	limit	i (i=i+1)	i < limit	String
1	10	0	0<10	...
2	10	1	1<10	...
3	10	2	2<10	...
⋮	⋮	⋮	⋮	
10	10	9	9<10	...
11	10	10	~~10<10~~	

Date

Q) Write table

```
int n;num;
printf("Enter num");
scanf("%d", &num);
for(int i=1, i<=n; i++)
{
printf("%d x %d = %d", num, i, num*i);
}
```

```
for (int i=1; i<=10, i++)
{
    printf("2 x %d = %d", i, 2*i).
}
```

Q) Write a program which takes upper limit & lower limit as input from user then sum numbers in series (baundries inclusive).

Example input : lowerlimit = 5 Upper limit = 10

" Output : Sum = 45

LL = 2 UL = 7

Sum = 27

```
int L, U; Sum=0;
printf("Enter UL & LL");
scanf("%d %d", &L, &U);
For (L, U<=U, L++)
{
    Sum+= L
    printf("%d", &Sum);
}
printf("%d", &Sum);
```

```
for(i=LL; i<=UL; i++)
{int Sum=0
    Sum = Sum + i
    print (newline)
    print (sum)
}
print (sum).
```

2
3
4
5
6
7
8
9
error.

m x n = no. of iterations.

Nested (For) Loop:

Date

2 for (i=1, i<=n, i++)

Outer

for (i → n)

3 for(j=1, j<=m, j++)

{ inner For (j → m)

{

}

}

i	j
1	1
	2
	3
2	1
	2
	3

$i = 1 \rightarrow n/4$

$j = i \rightarrow m/4$

Date

outer	inner	
1	* * * *	(4)
2	* * * *	(3)
3	* *	(2)
4	*	(1)

$j = m; j <= i$

inner loop dependent on outer loop.

outer	inner
1	*
2	* *
3	* * *
4	* * * *
⋮	
n	

```
for(i=1; i<=n; i++)
{
  for(j=1; j<=n; j++)
  {
    if (j==1 || j==n || i==0 || i==n)
      print ("*")
    else
      print (" ")
  }
  print (/n)
}
```

```
for (i=1; i<=n; i++)
{
  for (j=1; j<=i; j++)
  {
    print ("*")
  }
  print ('/n')
}
```

```
for (i=1; i<=n; i++)
{
  for (j=m; j>=1; j--)
  {
  }
}
```

```
for(i=1; i<=n; i++)
{
  For(j=1; j<=n; j++)
  {
    if (j==1 || j==n
        i==1 || i==n
        i==j)
  }
}
```

(i,j) 1,1 1,2 1,3 1,4 1,5 1,6

2,1 2,2 2,3 2,4 2,5 2,6

3,1 3,2 3,3 3,4

4,1 4,3 4,4

5,1 5,2 5,1

6,1 6,6

1,6 1,5 1,4 1,3 1,2 1,1

2,6 2,5 2,4 2,3 2,2 2,1

⋮

6,6

(i+j == n+1)

```
*                    1,1
* *                  2,1  2,2
*   *                3,1  3,2  3,3
*     *              4,1  4,2  4,3  4,4
* * * * *            5,1  5,2  5,3  5,4  5,5
```

Date

```
for (i=1 ; i<=n ; i++)
{
    For (j=1 ; j<=i ; i++)
    { if (j==1 ; i==n ; i==j
    }
}
print(/n)
}
```

```
0 0 0 0 1
0 0 0 1 1
0 0 1 1 1
0 1 1 1 1
1 1 1 1 1
```

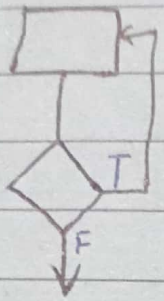
i	j	k	n
1	1 2 3 4	1	5
2	2 3 4	1 2	5
3	3 4	1 2 3	5
4	4	1 2 3 4	5
5	-	1 2 3 4 5	5

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j++)
    {
        print("0")
    }
    for(k=1; k<=i; k++)
    {
        print("1")
    }
    print(/n)
}
```

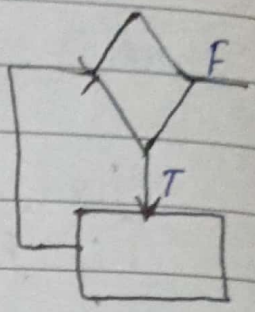
Date

Conditional loops: Do...while & while

```
Do
{
  statements
}
while(condition);
```




```
While (condition)
{
}
```



Q) Write a program which takes numbers input from user until user press x. Sum those number & print sum as output.

```
int n; sum=0   char ch;
Print ("Enter number")
scanf ("%d", &n);

while (n != 'x')
{ sum += n
  Print (Enter number)
  scanf (%d, n)
}

Print (Sum);
```

→ Static

Date

Arrays (containers) (works on index)

A collection of same data type variable.

Array will act store in continous memory location

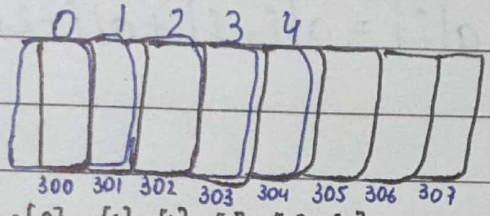
Static data type (predefined).

DataType name [No. of elements].

Absolute

offset → Relative addressing (if reference mentioned)

int a[5]



a is the reference Point Jaha se start hoti

Block + offset = Address

a

300 + 0 = 300

300 + 1 = 301

300 + 2 = 302

300 + 3 = 303

300 + 4 = 304

300 + 5 = 305

```
scanf("%d", a[i])

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

Q) Write a program which calculates sum of elements of array given by user.

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

size < 5

0	1	2	3	4
13	5	6	8	10

a_0

a_i a[i]

a_2

a_3

a_4

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

Date

0	1	1	2	3	5	8	...
a_0	a_1	a_2	a_3	a_4	a_5	a_6	

$a_2 = a_1 + a_0$

$a_3 = a_2 + a_1$

$a_4 = a_3 + a_2$

$a_5 = a_4 + a_3$

$a_6 = a_5 + a_4$

```
a[0] = 0
a[1] = 1
for(int i=2; i<Size; i++)
{
    a[i] = a[i-1] + a[i-2]
}
```

```
for(int i=0; i<Size-2; i++)
{
    a[i+2] = a[i+1] + a[i]
```

2	1	0
3	2	1
4	3	2
5	4	3
6	5	4

CamScanner

```
for(int i=0; i<3; i++)
{ for(int j=0; j<2; j++)
  { for(int k=0; k<4; k++)
    { sum+=a[i][k]*b[k][j];
    }
    result[i][j] = sum;
    sum = 0;
```

Matrix Multiplication

Date

Matrix Addition ;]

Scalar Multiplication on 2D a[0][0] = 2*a[0][0] / a[i][j] = num × a[i][j]

Matrix Multiplication

row processing (row fixed) a[r][c] r: 0,1,2,3 c: 0,1,2,3

Col processing (col fixed) a[r][c] r: 0,1,2,3 c: 0,1,2

	0	1	2	3
0	6	9	15	3
1	1	2	5	7
2	8	13	4	19
3	26	30	45	10

2*a[0][0]

Matrix Addition:

a[3] = [15, 20, 16]

b[3] = [1, 2, 1]

c[ ] = [ ]

$C_0 = a_0 + b_0$

$C_1 = a_1 + b_1$

$C_2 = a_2 + b_2$

```
for(i=0; i<n; i++)
{
  c[i] = a[i] + b[i]
}
```

12	18	30	6
2	4	10	14
16	26	8	38
52	60	40	100

```
for(i=0; i<row; i++)
  for(j=0; j<col; j++)
```

a[i][j] = num*a[i][j]

a

00	01
15	20
16 (10)	17 (11)

b

00	01
1	2
3 (10)	4 (11)

$C_{00} = a_{00} + b_{00}$

$C_{01} = a_{01} + b_{01}$

$C_{10} = a_{10} + b_{10}$

$C_{11} = a_{11} + b_{11}$

```
for(i=0; i<n; i++)
  for(j=0; j<m; j++)
    c[i][j] = a[i][j] + b[i][j]
```

Matrix Multiplication:

a[ ][2] × b[2][ ]

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

a b

$$[2 \quad 1] \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

a b

$$\begin{bmatrix} 4 \\ 2 \end{bmatrix} [2 \quad 1]$$

c[0][0] row0 × col0

$C_{ij} = a_{i0} b_{0j} + a_{i1} b_{1j}$

cola == rowb

bkacol, akirow

00	01
ae + bg	af + bh
ce + dg	cf + dh
10	11

$C_{00} = a_{00} b_{00} + a_{01} b_{10}$

$C_{01} = a_{00} b_{01} + a_{01} b_{11}$

$C_{10} = a_{10} b_{00} + a_{11} + b_{10}$

$C_{11} = a_{10} b_{01} + a_{11} b_{11}$

akirow, bkacol

2 × (1) (1) × 2

1 × (2) (2) × 1

a b

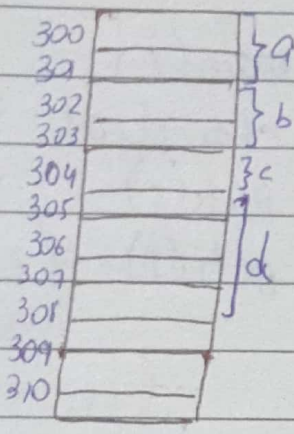
$$[2 \quad 1] \begin{bmatrix} 4 & 1 \\ 2 & 2 \end{bmatrix}$$

1 × (2) (2) × 2

CamScanner

Date

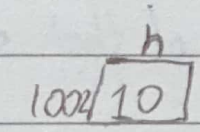
int a
int b
char c
float d



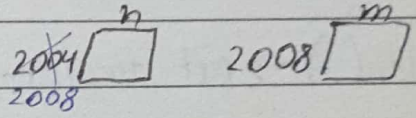
Variable = store values
Pointer Variable = store address

* represents pointer eg: *a
int *a;
int *a=0; (Null)

int n=10
int *n



int *n
int *m



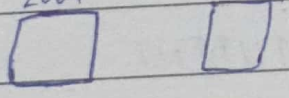
n = m

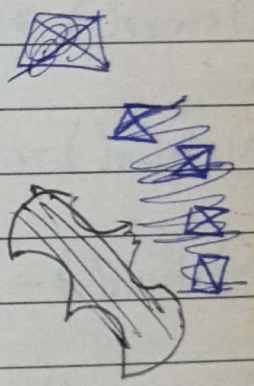
print("%d", n) → 5
print("%d", &n) → 2004
print("%d", *p); → 5
print("%d" p); → 2004

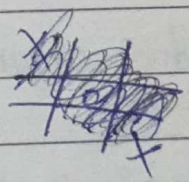
p = &n

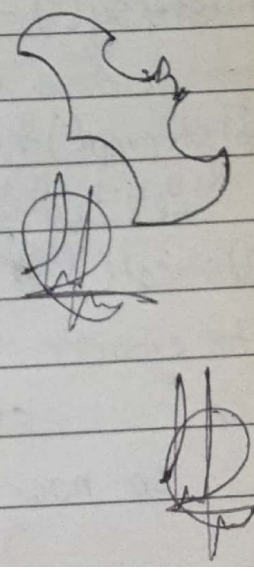
Sum(&a, &b)
↓
Address
2004 (Pointer)

2004 4002
Sum(int *x, int *y)
temp = *x
x = y
*y = temp
x = 4002









scanf → space, tab, enter so we use gets to take input

Strings

Date

```
#include<conio.h>
main(){
char s[10]={'A','N','A','S',' ','\0'}
for(i=0; i!='\0'; i<=4; i++)
printf("%c", s[i])
```

```
main(){
char s[20]
puts(s)        printf("%s", s)
gets(s)   /    scanf("%s", s)
```

Strings related pre-defined functions:

① strlen( ) → put array/string to find the length of that string.

② strrev( ) → to reverse string

③ strlwr( ) → to convert those letter which are in uppercase

④ strupr( ) → convert to uppercase

⑤ strcpy( ) → need 2 arguments (to copy string) for eg: strcpy(s, "KHAN

⑥ strcmp( ) → to compare 2 string strcmp(S1 "KING", S2 "BILAL") it will return ASCII code of different if it return 0 zero so value it some.
↳ 0, 1, -1 if 1 (so S1>S2)
↳ Both are equal if -1 (so S1<S2)

⑦ strcat( ) → To add something in string (Concatination)
for example we have string s contain HELLO
strcat(s, "student")
so now so contain HELLOSTUDENT.

length determined by first NULL in the string

DSU\0 SUFFA
3
DSUSUFF\0A
7

31) Array (String)

Date

```
main(){
char s[3][10]
int i;
printf("Enter 3 string")
~~gets(s[0][0]~~
for(i=0; i<=2; i++)
  gets(s[i])
for(i=0; i<=2; i++)
  printf("%s\n", s[i]);
}
```

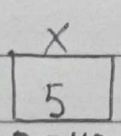
Output

ANAI
ALI
RAFAY

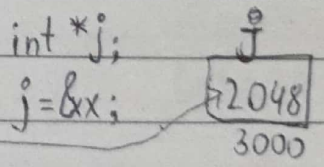
•) For address printing we use %u

•) Pointer always contain 2 bytes in memory.
↳ special variable created to store address.

int x = 5
&x = 7
↳ it wrong
as we can't assign any constant to address as it's not a variable



int *j;
j = &x;

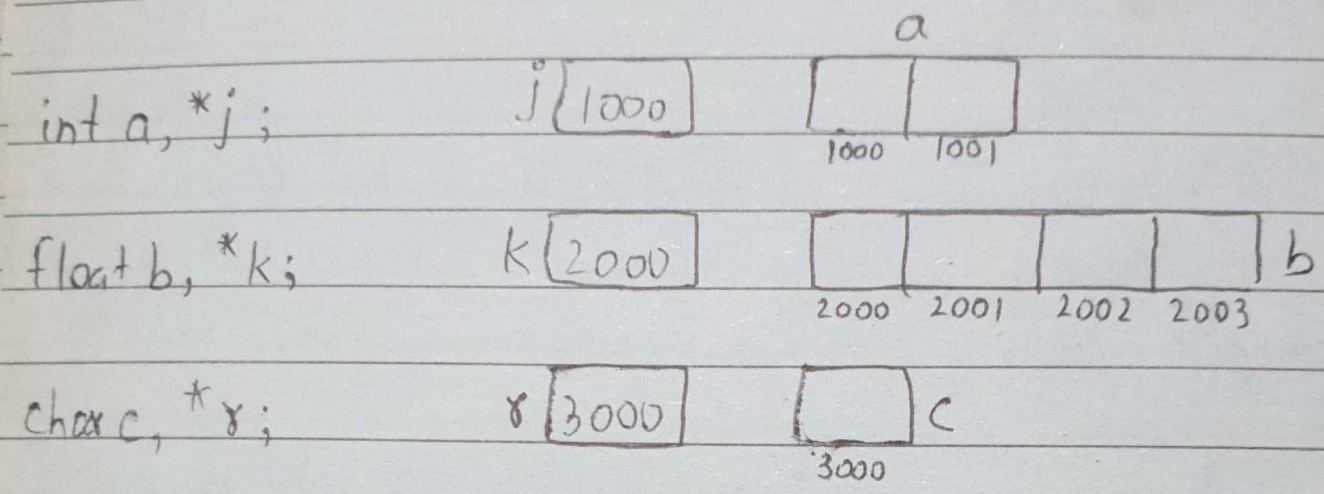



```
main(){
int x=5, *j;
j=&x;
printf("%d%u", x, j);      5 2048
printf("%d %u", *j, &x);   5 2048
printf("%u", *&j)          2048
```

baseAddress = a = 1000
b = 2000
c = 3000

Date

Base Address



j = &a; k = &b; r = &c;

Caller Actual Parameter: The Parameter passed to a function (throw)
Callee Formal Parameter: The Parameter received by a function (catch)

Recursion:

Date _____

```
void Timer(int n)
{ if (n<1) → base
      return;
  else{
  print(value of n)
  Timer(n-1); } → recursive
}
    if n=5
Output: 5,4,3,2,1
```

Factorial

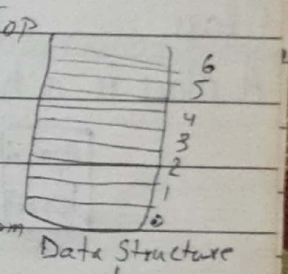
```
if (n≤1) → Base
else
n * F(n-1) → recursive
```

Power (exponential)

```
Pow(x,n)
if (x==0)
return;
else
x * Pow(x,n-1)
```

```
int plus_one(int x)
{
  return (x+1);
}
int plus_two(int n)
{
  return (plus_one(x+1));
}
```

```
int main()
{
result = plus_one(0);
result = plus_two(result);
print(result);
}
```



Stack

LIFO

Last in First out.

insert → Push

Delete → Pop.

Activation Record.