C --> Middle level language

categories of programming languages:

1. low level language
2. middle level "
3. high level "

Problem Solving

1. Write down problem
2. Write all possible solutions, find best one solution
3. Out of box thinking

SDLC --> Systems Development Life Cycle
--> Software Development Life Cycle

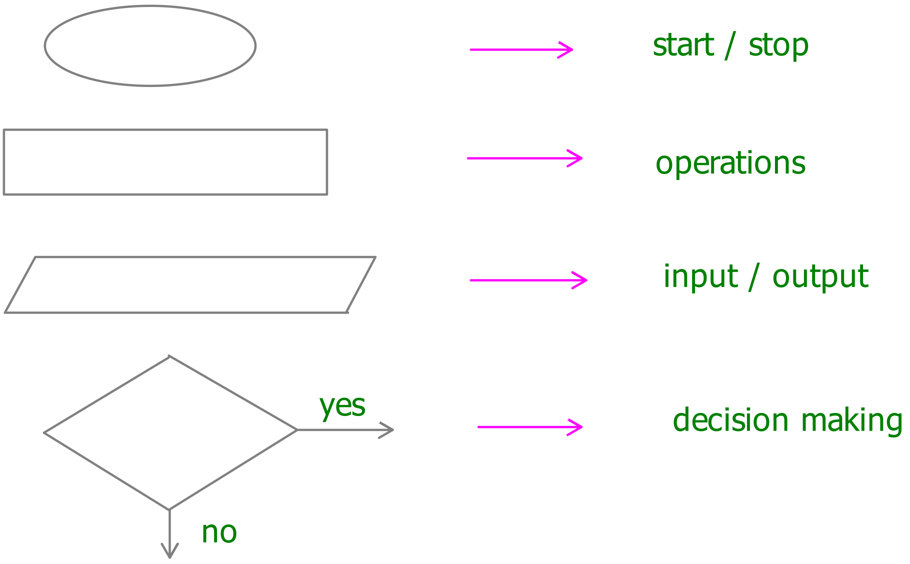
1. Adding 2 numbers:

Natural language:
1. start
2. read n1 and n2
3. add n1, n2 and store in sum
4. display sum
5. stop

Pseudo code:

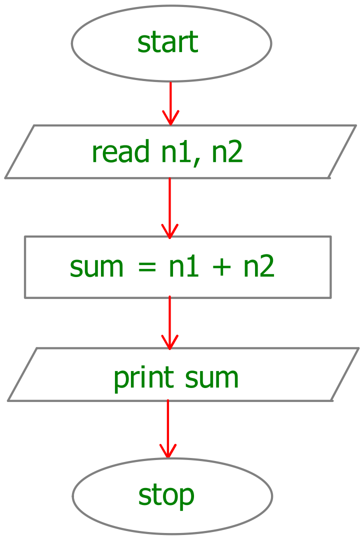
```
1. BEGIN
2. read n1, n2
3. sum = n1 + n2
4. print sum
5. END
```

Flow chart:



○ → connector

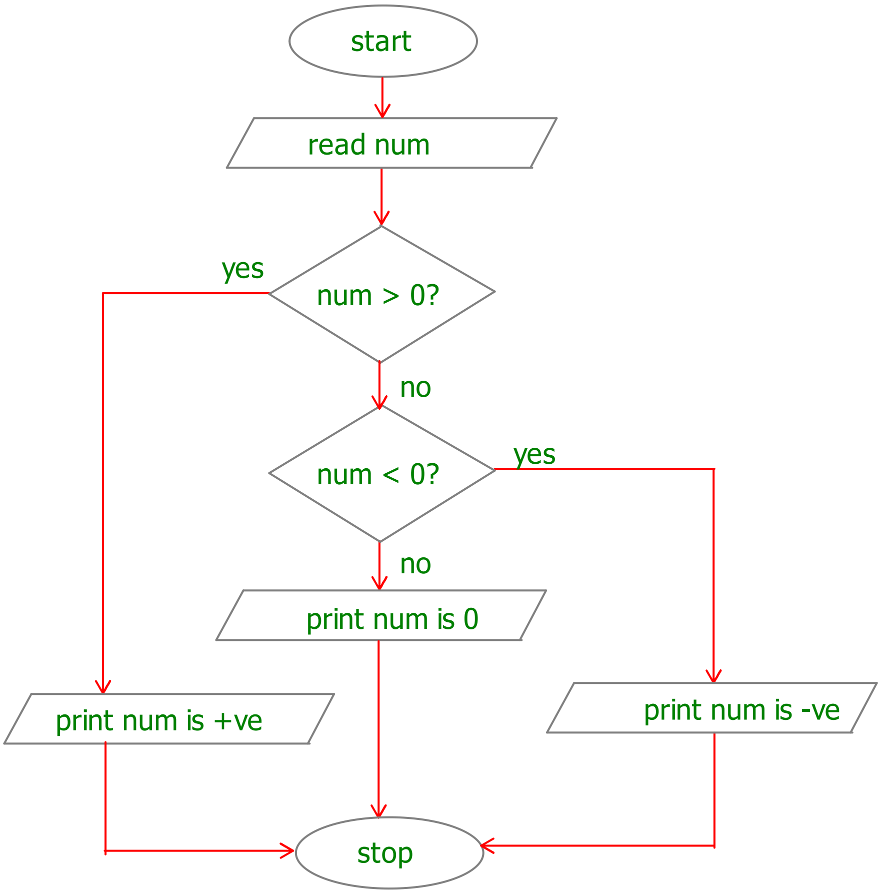
⟶ → flow



2. Algorithm to find entered number is +ve, -ve or 0

```
Natural language
1. start
2. read num
3. comapre if num is greater than 0
        if yes, print num is +ve
    if not, compare num is less than 0
        if yes, print num is -ve
    if not, print num is neither +ve nor -ve
4. stop
```

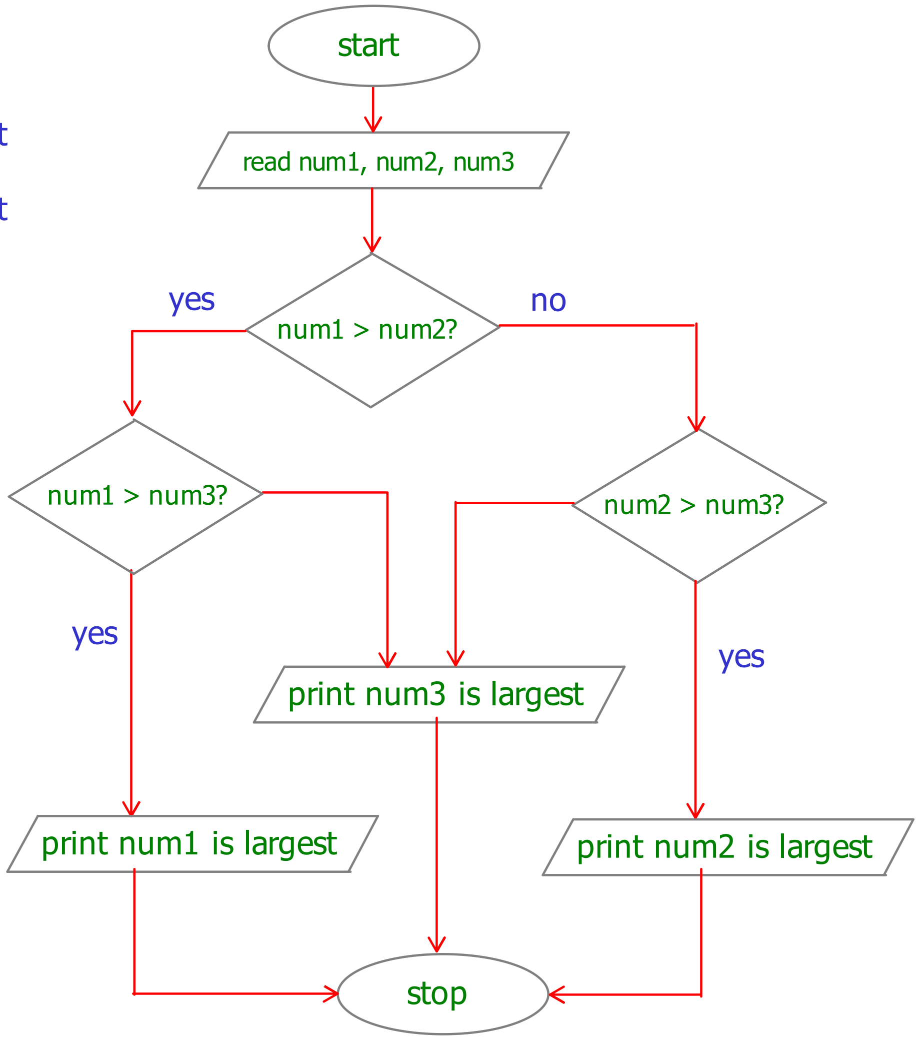
```
Pseudo code:
1. BEGIN
2. read num
3. if num > 0
        print num is +ve
    else if num < 0
        print num is -ve
    else
        print num is 0
4. END
```



3. Algorithm to find the largest of 3 numbers:

Pseudo code:

```
1. start
2. read num1, num2, num3
3. if num1 > num2
        if num1 > num3
            print num1 is the largest
        else
            print num3 is the largest
   else if num2 > num3
        print num2 is the largest
   else
        print num3 is the largest
4. stop
```



4. Algorithm to find the sum of 1 to n integers:

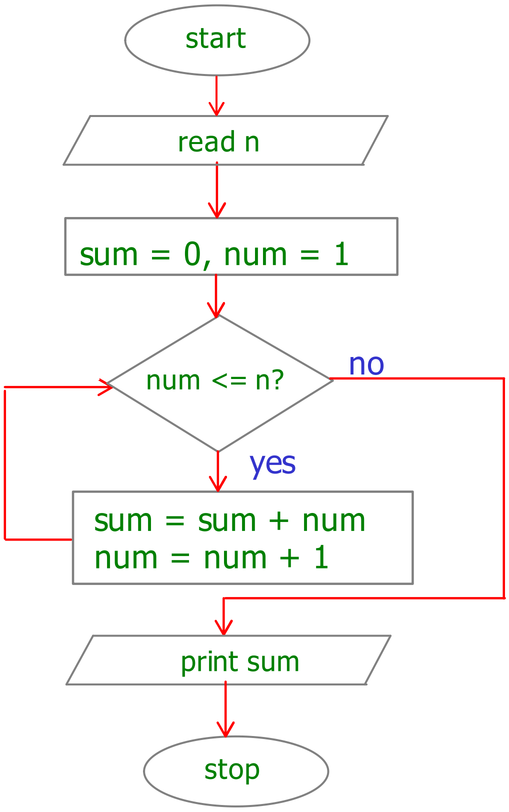
Pseudo code:

```
1. start
2. read n
3. declare sum = 0, num = 1
4. loop: num <= n
         sum = sum + num
         num = num + 1
5. print sum
6. stop
```

n = 5

sum = 1 + 2 + 3 + 4 + 5

for, while, do-while



5. Algorithm to find the sum of n integers:

Pseudo code:

```
1. start
2. read n
3. declare sum = 0, count = 1, num
4. loop: count <= n
         read num
         sum = sum + num
         count = count + 1
5. print sum
6. stop
```

n = 5

10
12
20
8
15

15
65
6

6.

```
 1 2 3 4 5
|*|*|*|*|*|
```

row = 1
col = 5

```
1. start
2. read n
3. count = 1
4. loop: count <= n
         print "* "
5. print new line
6. stop
```

7.

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

```
1. start
2. read row
3. loop i: 1 to row
        loop j: 1 to row
                print "* "
        print new line
4. stop
```

i -> row
j -> col

8.

	1	2	3	4
1	*			
2	*	*		
3	*	*	*	
4	*	*	*	*

```
1. start
2. read row
3. loop i: 1 to row
        loop j: 1 to i
                print "* "
        print new line
4. stop
```

9.

	1	2	3	4
1	*	*	*	*
2	*	*	*	
3	*	*		
4	*			

```
1. start
2. read row
3. loop i: 1 to row
        loop j: 1 to (row - i + 1)
                print "* "
        print new line
4. stop
```

i <= row
j <= (row - i + 1)

// j: row to 1

```
loop i: 1 to row   i <= row
        loop j: row to 1          j = j - 1
                    j >= 1
```

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

10

	1	2	3	4	5	6	7
1				*			
2			*		*		
3		*		*		*	
4	*		*		*		*

```
1. start
2. read row
3. loop i: 1 to row
        loop j: 1 to (row - i)
                print " "
        loop k: 1 to i
                print "* "
        print new line
4. stop
```

row = 4
i = 3
    j = 1
    k = 1, 2, 3

			*			
		*		*		
	*		*		*	

low level language --> low abstraction
middle level language --> moderate abstraction
high level language --> high abstraction level

middle to high level conversion --> using functions
middle to low level conversion --> using pointers

```
/* ............................
..multi-line commenting
...............................
......................*/
```

```
// single line comment
```

```
#include<stdio.h>
        printf() -->print anything
        scanf() --> reading input
```

```
sudo apt-get update; sudo apt install gcc --> linux
```

Visual Studio Code

GCC --> GNU's Compiler Collection

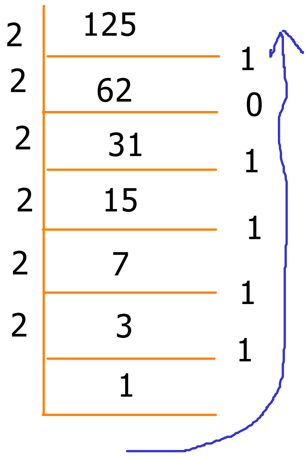
gcc file.c	a.out
gcc file.c -o hello.out	hello.out

Number System

1. Decimal	Base 10	0 - 9
2. Binary	Base 2	0, 1
3. Octal	Base 8	0 - 7
4. Hexadecimal	Base 16	0 - 9, A - F

1. Decimal to binary:

125 = 1111101



2	125	1
2	62	0
2	31	1
2	15	1
2	7	1
2	3	1
	1	

2. Binary to decimal:

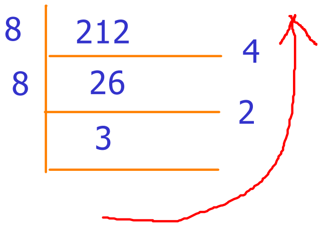
1111101

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

(1 x 2^6) + (1 x 2^5) + (1 x 2^4) + (1 x 2^3) + (1 x 2^2) + (1 x 2^0) = 125

3. Decimal to Octal:

212 = 324



8	212	4
8	26	2
	3	

4. Octal to Decimal:

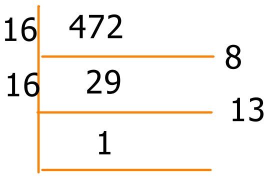
324

(3 x 8^2) + (2 x 8^1) + (4 x 8^0) = 212

A - 1010	10
B - 1011	11
C - 1100	12
D - 1101	13
E - 1110	14
F - 1111	15

5. Decimal to Hexadecimal:

472 = 1D8



16	472	8
16	29	13
	1	

6. Hexadecimal to Decimal:

1D8

(1 x 16^2) + (13 x 16^1) + (8 x 16^0) = 472

7. Binary to Octal:

7 --> 111 --> max is 3 digits

```
0 110 101 011 011 101 110 001
0   6   5   3   3   5   6   1
```

8. Octal to binary:

7250216 --> every digit represented in 3 bits

111 010 101 000 010 001 110

9. Binary to Hexadecimal:

F --> 1111 --> max 4 bits

```
10 1011 0010 1011 1101 0100
 2    B    2    B    D    4
```

10. Hexadecimal to Binary:

1AD540B2 --> represent every digit in 4 bits

0001 1010 1101 0101 0100 0000 1011 0010

11. Octal to Hexadecimal:

1) Octal to binary, binary to hexadecimal
2) Octal to decimal, decimal to hexadecimal

752104

111 101 010 001 000 100

```
0011 1101 0100 0100 0100
   3    D    4    4    4
```

12. Hexadecimal to Octal:

1F2C5D8

0001 1111 0010 1100 0101 1101 1000

```
0 001 111 100 101 100 010 111 011 000
0   1   7   4   5   4   2   7   3   0
```

Data Representation

1 byte

1	1	1	1	1	1	1	1

Nibble

Range of data in 1 byte:

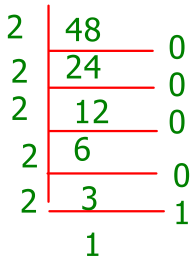
Decimal	0	255
Binary	0b00000000	0b11111111
Octal	00	0377
Hexadecimal	0x0	0xFF

'A' 'a' '@' ',' '0' '9'

character code --> 1 byte --> max 7bits

least --> 0
max --> 127

```
int num = 0;        00000000
char ch = '0';      00110000
```



13 in binary:

```
00000000 00000000 00000000 00001101        int num = 13;
```

-13 --> 2's complement of 13

```
 00000000 00000000 00000000 00001101        int num = -13;
 11111111 11111111 11111111 11110010 +
                                    1
 11111111 11111111 11111111 11110011        --> -13
```

char ch = -13;

```
00001101                                   char ch = 'a';
11110010 +                                 char ch = 97;
       1
11110011     --> 243        --> 256 - 13 = 243
```

-num = 2^n - num
n --> 8, if 1byte
n --> 32, if 4bytes