Problem Solving Through Programming in C

Tutorial Session 1

Prof. Anupam Basu
Dept. of Computer Science & Engg.
IIT Kharagpur

Siddhant Mohapatra
PMRF Scholar
IIT Madras

Problem Solving

- Any problem, which can be thought of in logical sequences can be solved using programming.
- "Logical thinking" – the most important part of the programming ← ability to do this
- How can you think logically?

e.g. print all factors of 50.

what is a factor?

what can be a factor of a no.? $1 \leq \text{factor} \leq 50$

50/1 50/3 } if its perfectly divisible, it is a factor of 50.
50/2

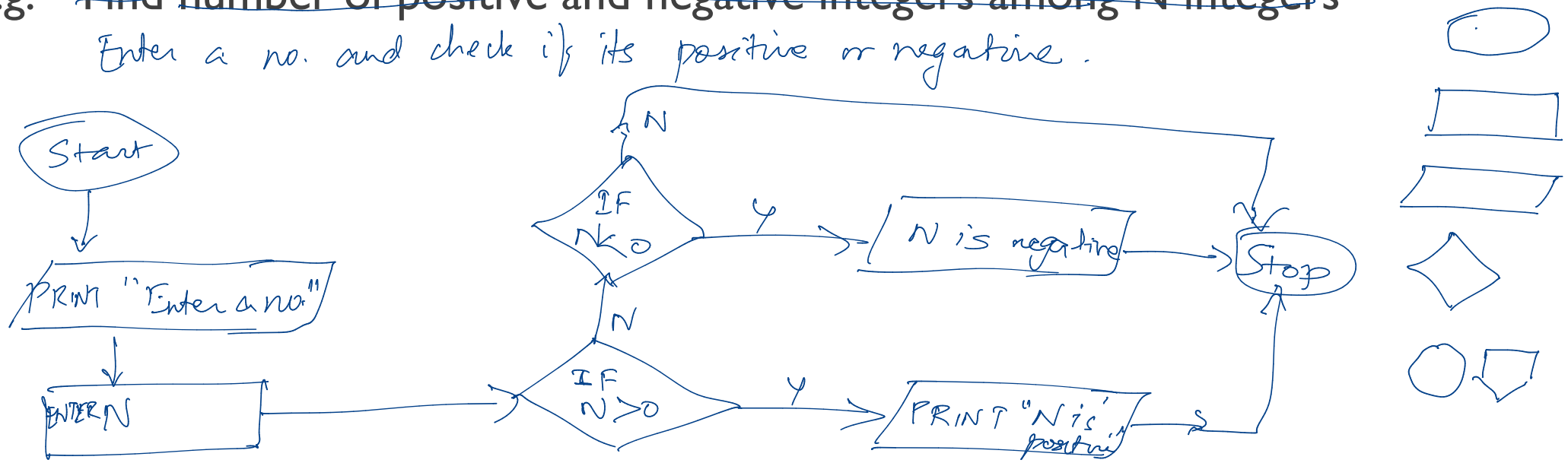
Logical thinking and algorithms

- What is an algorithm? - a set of instructions to solve a problem
- Different methods of explaining an algorithm: Flow charts, Pseudocode

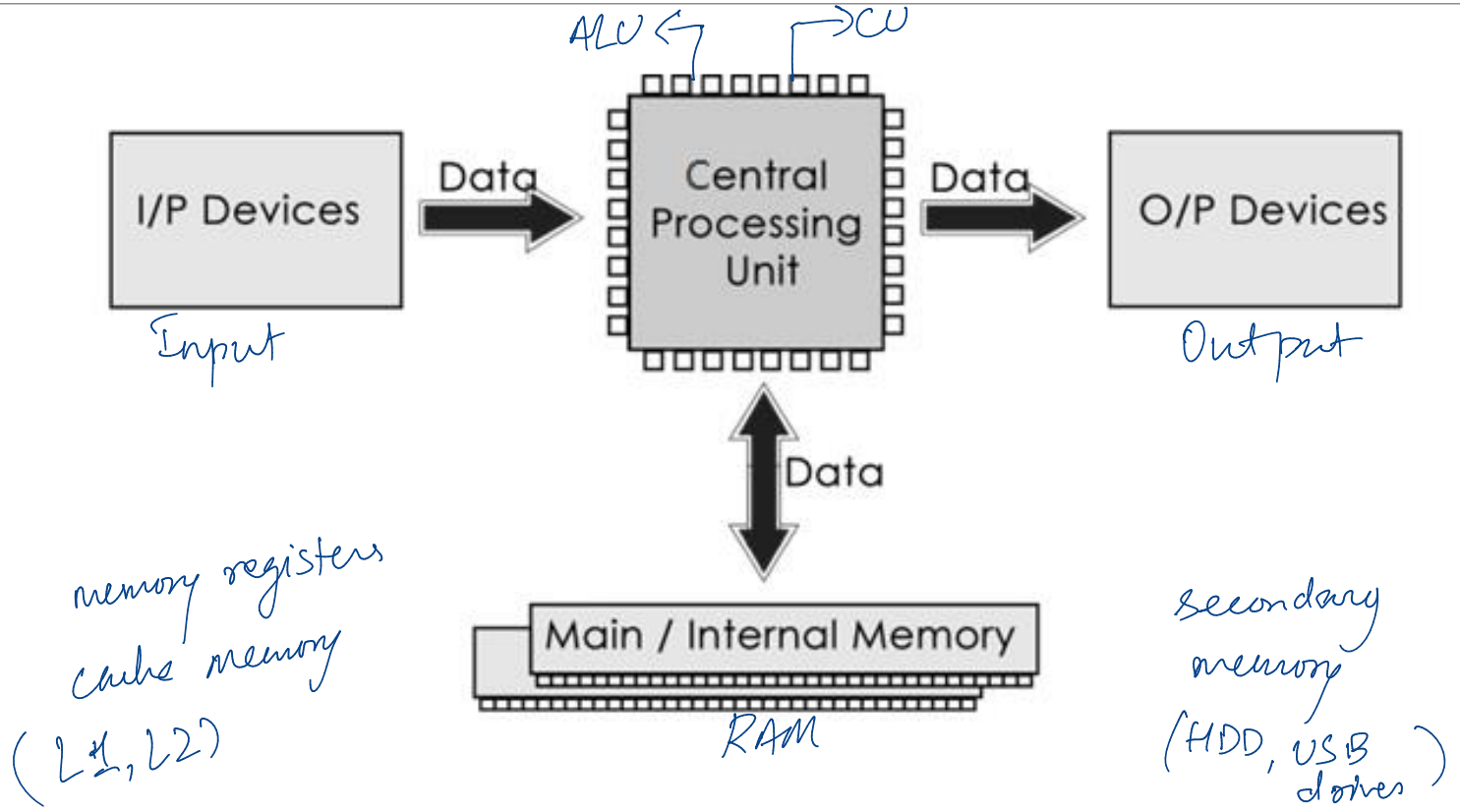
pictorial representation

e.g. – ~~Find number of positive and negative integers among N integers~~

Enter a no. and check if its positive or negative.



Components of a computer



Number Systems

- Binary – Base 2 system – 0 and 1 → computer uses this
- Octadecimal – Base 8 system - 0, 1, 2, 3, 4, 5, 6, 7
- Decimal – Base 10 system - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Hexadecimal – Base 16 system - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

10 11 12 13 14 15

commonly used

base 3 0, 1, 2

e.g. – Convert $(1011.101)_2$ to decimal

1 0 1 1 . 1 0 1

$2^3\ 2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}$

+

e.g. – Convert $(6A7F)_{16}$ to decimal and binary

6 A 7 F

$16^3\ 16^2\ 16^1\ 16^0$

+

( ) $_{10}$ decimal

2 | 56
2 | 28 – 0
2 | 14 – 0
2 | 7 – 0
2 | 3 – 1
2 | 1 – 1
0 – 1

$(111000)_2$ = $(56)_{10}$

Programming Language

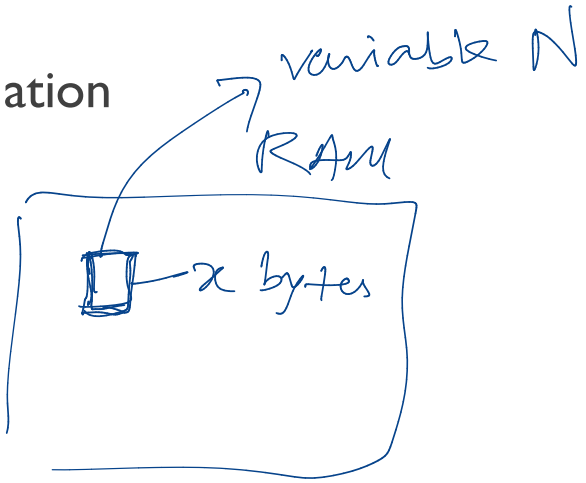
- A method of conveying the instructions to the computer
- 3 levels of languages:
  - Machine level (Compiled/interpreted code)
  - Assembly level
  - High level (Source code)

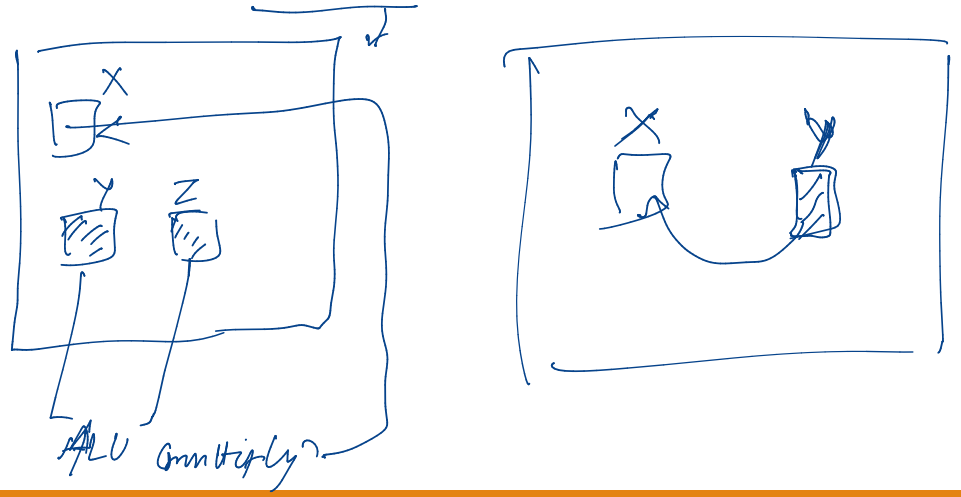
compiler / interpreter

- Compiler/ Interpreter – converts the source code to machine readable form

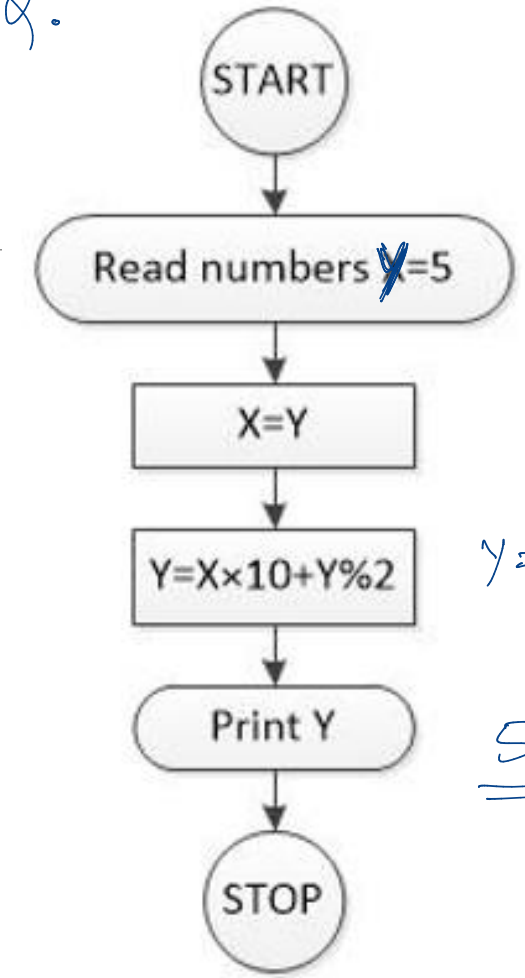
C: Memory and Variables

- Memory: Primary, secondary, others (cache, registers, buffers)
- 1 Byte = 8 Bits (Bit is the smallest unit of memory)
- Variables – memory locations in the RAM which is allocated during compilation
- What does X = Y mean?
- What happens when we write X = Y*Z ?





8.



~~X=5~~ Y=5

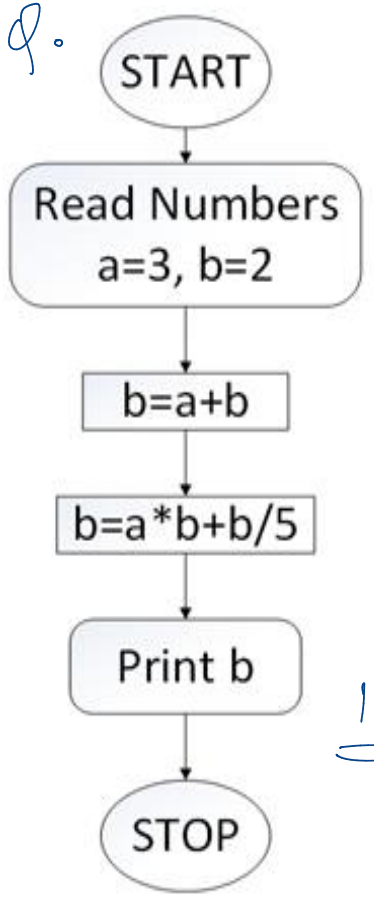
X = 5

Y = 5×10+5%2 = 50+1 = 51

(% → remainder)

51

9.



a = 3
b = 2

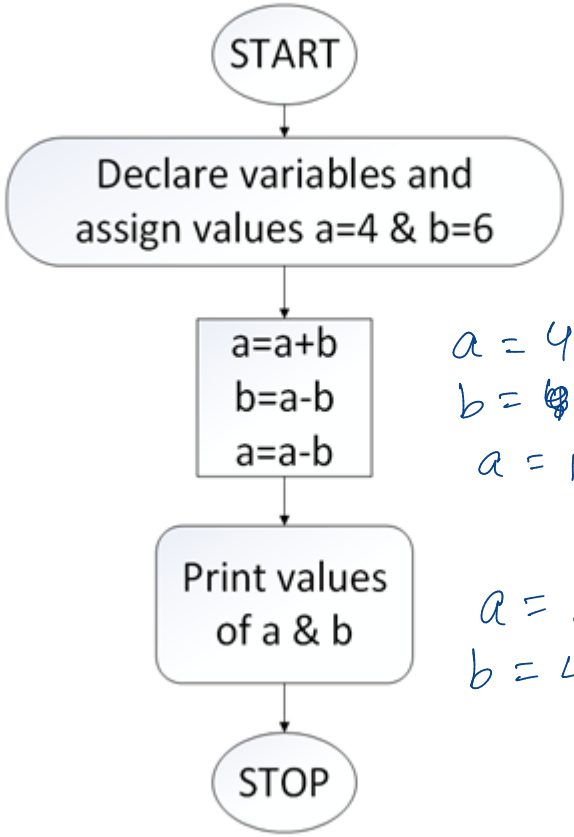
b = 3+2 = 5

b = 3*5+5/5
= 15+1 = 16

16

a) 4
b) 8
c) 16
d) 20

Q. The print values of 'a' and 'b' of the flowchart below are



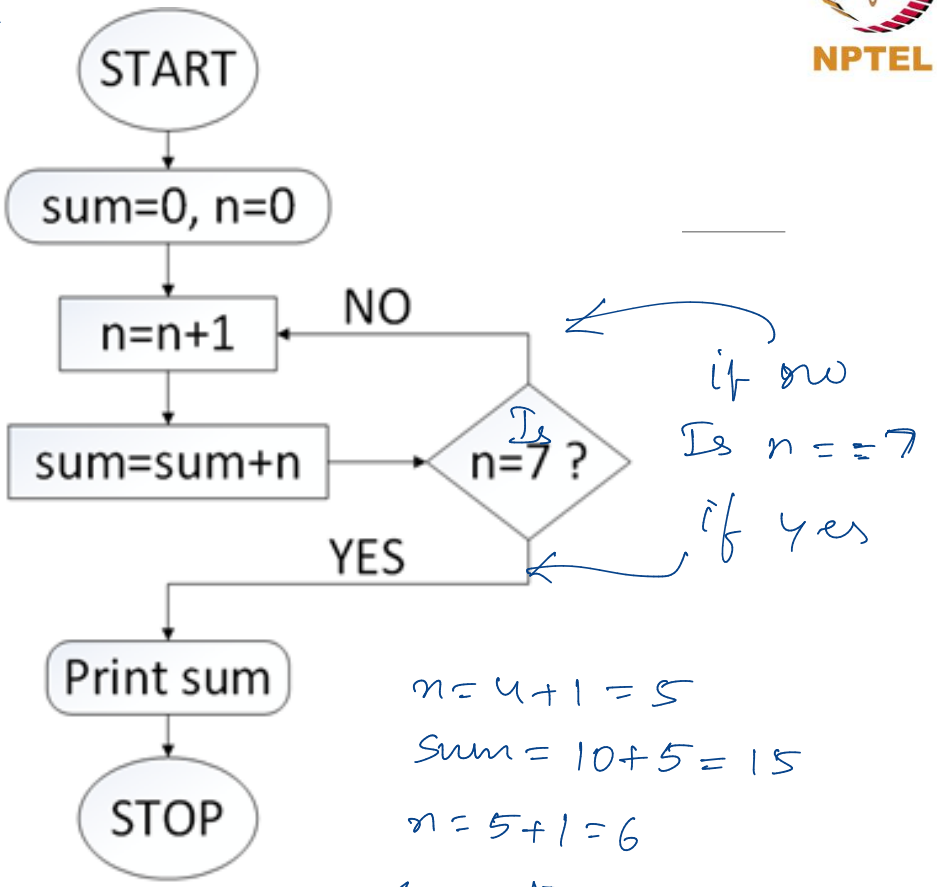
a = 4
b = 6

a = 4+6 = 10
b = 10-6 = 4
a = 10-4 = 6

a = 6
b = 4

a) a=4, b=6
b) a=6, b=4
c) a=10, b=2
d) a=2, b=10

Q.



sum = 0
n = 0

n = 0+1 = 1
sum = 0+1 = 1
n = 1+1 = 2
sum = 1+2 = 3
n = 2+1 = 3
sum = 3+3 = 6
n = 3+1 = 4
sum = 6+4 = 10

n = 4+1 = 5
sum = 10+5 = 15
n = 5+1 = 6
sum = 15+6 = 21
n = 6+1 = 7
sum = 21+7 = 28
Is n=7 yes
sum

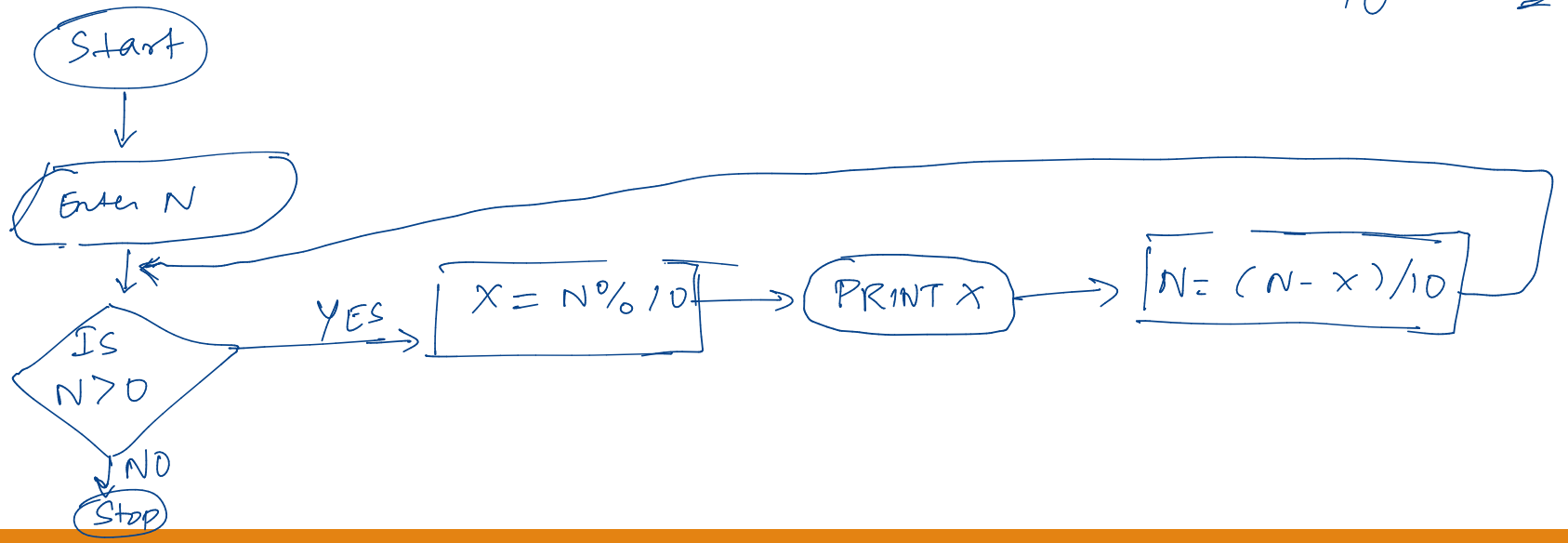
a) 21
b) 28
c) 30
d) 40

Q. Draw a flowchart to extract all digits from a natural number.

$N = 756$

$756 \% 10 = 6$ $\quad 75 \% 10 = 5$ $\quad 7 \% 10 = 7$

$\frac{756-6}{10} = 75$ $\quad \frac{75-5}{10} = 7$ $\quad \frac{7-7}{10} = 0$

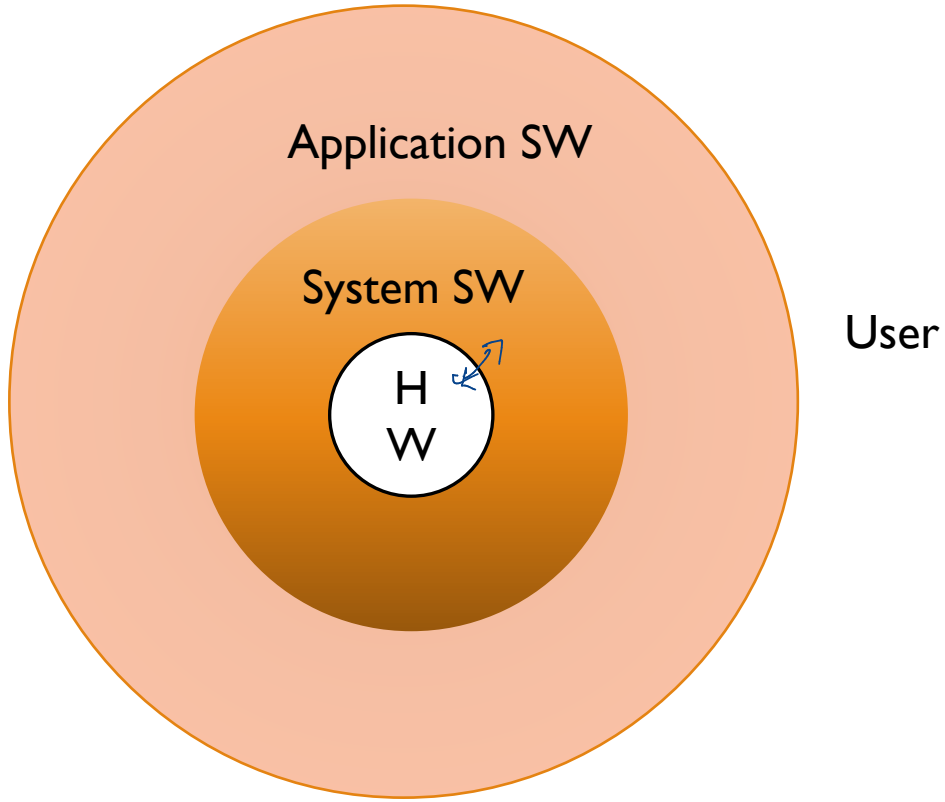


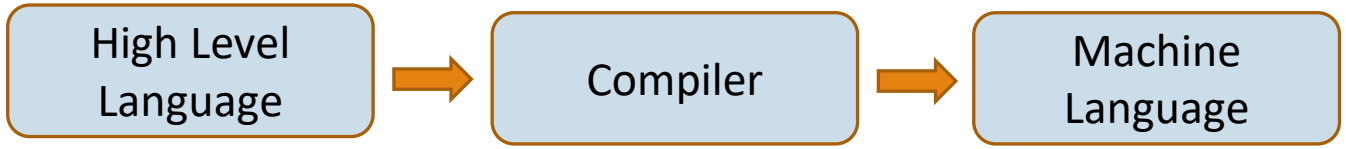

```
enter N
while N>0
    x = N%10
    print x
    N = (N-x)/10
goto
```

```
10 enter N
20 if N>0
30 x = N%10
40 print x
50 N = (N-x)/10
60 goto 20
```

Types of software

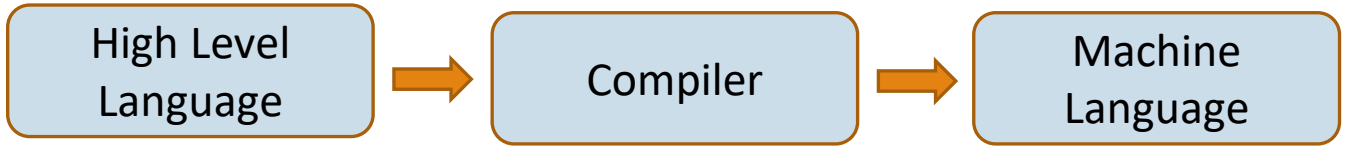
- Application software → IDE Codeblocks
- System software
  - Compilers, interpreters
  - OS
    - Single user
    - Multi user
      - Time sharing (multi-user)
      - Multitasking (multi-tasks)





Syntax and variables

- Variables – memory locations in the RAM which is allocated
- Naming of variables
  - string of alphanumeric characters with some allowed special characters (underscore)
  - Shouldn't start with a number
  - Shouldn't be a keyword → special purpose in C
  - Is case-sensitive — Case, CASe, case are different variables
  - Recommended not to start with an underscore



Variable names beginning with underscore is not encouraged in C. Why?

a) It is not standardized
b) To avoid conflicts since assemblers and loaders use such names
c) To avoid conflicts since library routines use such names
d) None of these

Which of these is an invalid variable name?

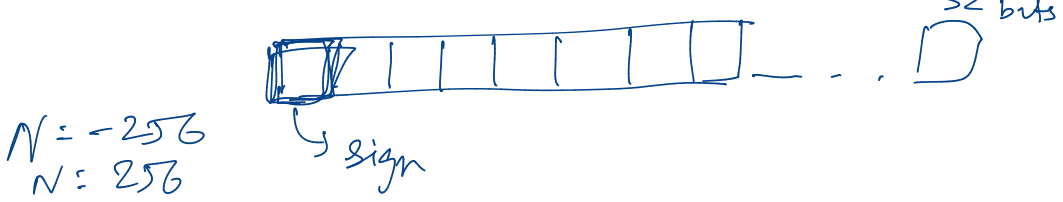
a) `pa_123x1`
b) `P123x_a`
c) `748_pax`
d) `_pax748` — not recommended

Structure of a C program

- Needs to have a main() function and can have more functions.
- Has an in-built library of functions for various operations.

Q. In case of signed int variables:

a) The leftmost bit is reserved for sign
b) The rightmost bit is reserved for sign
c) No bit is reserved for sign
d) None of the above



Q. What will be the output of the program given below?

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

Compilation error

Q. What is the value of x in this C code?

```
#include <stdio.h>
int main()
{
   int x = 4 *5 / 2 + 9;
   return 0;
}
```

a) 1
b) 17
c) 19
d) 16

Data types and print formatting

- Control character: %d, %f, %2.3f, %c and so on
- Escape sequence: '\n', '\t', '\\', '\b' and so on

Q. What is the output of the following snippet?

```
#include <stdio.h>
int main(){
    float b = 57;
    printf("%f", b);
    return 0;
}
```

57.000000

6 zeros

Q.

```
#include <stdio.h>
int main()
{
    float a = 4512.6526;
    printf("%.3f", a);
    return 0;
}
```

What is the output of the above?

a) 4512
b) 4512.6
c) 4512.652
d) 4512.653 ✓

x = 50.2345
printf("%.2f", x); => 50.23

int n = 50;
printf("%d", n);

printf("%d\n", n);

50*

%.3f → 3 digits after decimal

or

3 significant digits after decimals?

4512.653

%10.3f => __4512.653

↳ at least 10 characters long

Macros

#define a 6
const int a=6;
keyword
Constants

Q. What is the output of the following program?

```
#include <stdio.h>
#define a 6
int main()
{
        int a=3;
        a = a+1;
    printf("%d", a);
    return 0;
}
```

redefining a constant macro (a) value of 6
already a macro
macro can't be redefined within a function

a) 6
b) 3
c) 4
d) Compilation error ✓

Q. What is the output of the following code?

```
#include<stdio.h>
#define fun(x) (x*x-x)
void main()
{
  float i;
  i = 37.0/fun(2);
  printf("%.2f", i);
}
```

macro. fun(x) (x*x-x)

37.0/(2*2-2)
= 37.0/2
= 18.50

18.50

Miscellaneous

Q. What is the output of the following C code?

```
#include <stdio.h>
int main()
{
        int var = 0110;
        var=var+7;
        printf("%d", var);
        return 0;
}
```

What will be the output of the above?

a) 106
b) 117
c) 79
d) 12

Q. What is the value of b after execution of the following C code?

```
#include <stdio.h>
int main()
{
int a = 7, b=10;
a = b;
b = a;
return 0;
}
```

%f - placeholder for a floating point value

" → user-defined header files
> → pre-defined header files
} usual notation

Miscellaneous

int x = {1, 2, 3, 4}
value x = ?
of
Q4, 8

Q. What will be the output of the following C code?
[N.B: - .2f is used to print up to 2 decimal places of a floating-point number]

```
#include <stdio.h>
int main()
{
float a = 7.0;
printf ("The output is  %.2f", (13/5)*a + 10);
return 0;
}
```

14 of 10 = 24.00
2 * 7.0
Both are integers
it will give you an integer.
"%.2f"
24.000000

a) 28.20
b) 21.00
c) 24.00
d) 21.20

Q. What will be the output?

```
#include <stdio.h>
int main() {
int x = 1, y = 3;
int t = x;
x = y;
y = t;
printf("%d %d", x, y);
return 0;
}
```

x=1, y=3
t=1
x=3
y=1

a) 1 3
b) 3 1
c) 1 1
d) 3 3