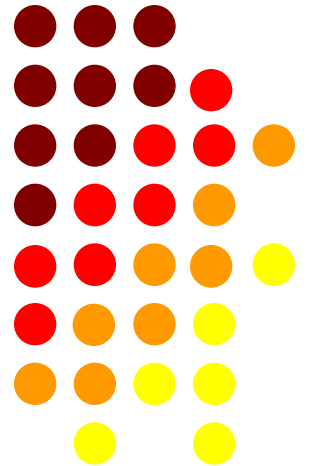


# Programming For Problem Solving

## Lecture 1



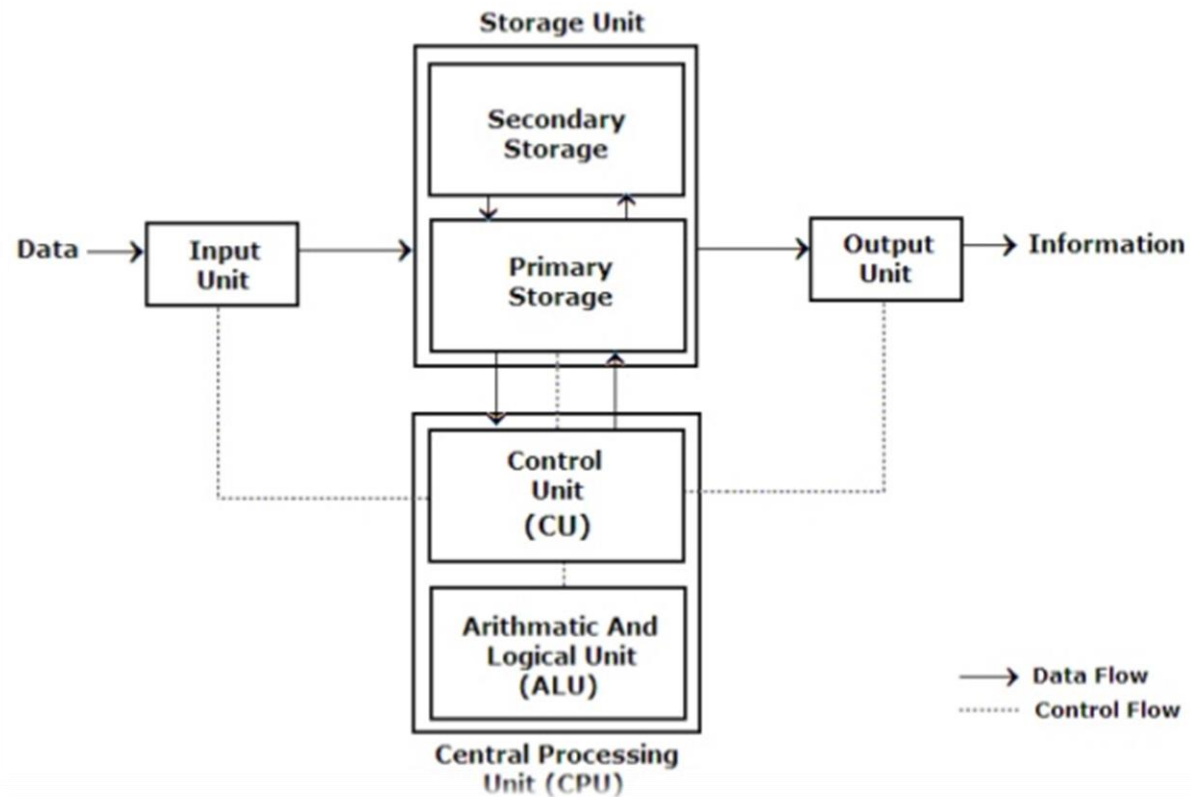
# Introduction to Computer

- The literal meaning of computer is, a device that can calculate. However, modern computers can do a lot more than calculate.
- **Computer** is an electronic device that receives input, stores or processes the input as per user instructions and provides output in desired format.



# Block Diagram of Digital Computer

## Block diagram of computer



# Components of Digital Computer

- A digital computer is considered to be a calculating device that can perform arithmetic operations at enormous speed.
- **1. Input Unit:** The commonly used input devices are keyboard, mouse etc. Thus, we can conclude that, all the input devices accept the data and instruction from outside world, convert it to a form that the computer can understand, supply the converted data to the computer system for further processing.
- **2. Storage Unit:** The storage unit of a computer holds data and instructions that are entered through the input unit, before they are processed. It stores programs, data as well as intermediate results and results for output. Its main function is to store information.



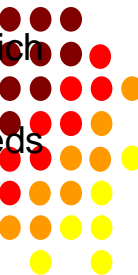
# Components of Digital Computer (Cont..)

- The various storage devices can be divided into 3 main categories:

**(a) Primary Storage (Main Memory):** This memory is generally used to hold the program being currently executed in the computer, the data being received from input device, the intermediate and final results of a program. The primary memory is temporary in nature. The data is lost when the computer is switched off.

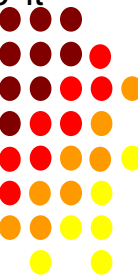
**(b) Secondary Storage (Auxiliary Memory):** It is a mass storage memory, slower but cheaper. It is non-volatile in nature i.e. data is not lost even if the power supply is switched off. Some of the most commonly used secondary storage devices are Hard Disk, Pen Drive etc. Their access time is in milliseconds.

**(c) Cache Memory (High Speed Buffer):** It is a high speed, expensive memory unit, which is placed between the processor & primary memory, to reduce the mismatch between the speeds of two units.



# Components of Digital Computer (Cont..)

- **3. Central Processing Unit (CPU):** The control unit and arithmetic logic unit of computer are together known as central processing unit (CPU). The CPU is like brain and performs following functions: It performs all calculations, it takes all decisions, and it controls all units of a computer.
  - (a) Control Unit:** It manages and coordinates the entire computer system and synchronizes its working, thus referred to as “Central Nervous System” or “Brain of the Computer”.
  - (b) Arithmetic and Logic Unit:** The function of an Arithmetic logic unit (ALU) is to perform arithmetic and logical operations such as addition, subtraction, multiplication, division, AND, OR, NOT, Exclusive OR etc.
- **4. Output Unit:** An output unit performs the reverse operation of that of an input unit, so it supplies information obtained from processing to outside world.



# Classification of Computer

S. No	Computer Type	Features
1	<b>Super Computer</b>	<ul style="list-style-type: none"> <li>• The fastest and most powerful type of computer is Supercomputers.</li> <li>• They are very expensive and are employed for specialized applications that require immense amounts of mathematical calculations.</li> <li>• For example, weather forecasting, nuclear energy research, and petroleum exploration.</li> </ul>
2	<b>Mainframe Computer</b>	<ul style="list-style-type: none"> <li>• It is a very large and expensive computer capable of supporting hundreds, or even thousands, of users simultaneously.</li> <li>• For example, online reservation system, online banking system, In hierarchy mainframes are just below supercomputers.</li> </ul>



# Classification of Computer (Cont..)

3	<b>Mini Computer</b>	<ul style="list-style-type: none"> <li>• In size and power, minicomputers lie between workstations and mainframes.</li> <li>• But in general, a minicomputer is a multiprocessing system capable of supporting from 4 to about 200 users simultaneously.</li> </ul>
4	<b>Workstations</b>	<ul style="list-style-type: none"> <li>• It is a terminal or desktop computer in a network. In this context, workstation is just a generic term for a user's machine (client machine) in contrast to a "server" or "mainframe." (Workstations lie between mini computer and personal computer).</li> </ul>
5	<b>Micro Computer or Personal Computer</b>	<ul style="list-style-type: none"> <li>• Desktop Computer: It is a personal or micro-mini computer sufficient to fit on a desk.</li> <li>• Laptop Computer: It is a portable computer complete with an integrated screen and keyboard.</li> </ul>





# Hardware & Software

Basis	Hardware	Software
<b>Definition</b>	Devices that are required to store and execute (or run) the software.	Software is a collection of program that enables a computer to perform a specific task
<b>Types</b>	Input, storage, processing, control, and output devices.	System software, Programming software, and Application software.
<b>Examples</b>	Monitor, Printer, Keyboard, Mouse, Scanners	Adobe Acrobat, Google Chrome, Microsoft Word, Microsoft Excel
<b>Durability</b>	Hardware wears out over time.	Software does not wear out over time.
<b>Nature</b>	Hardware is physical in nature.	Software is logical in nature.



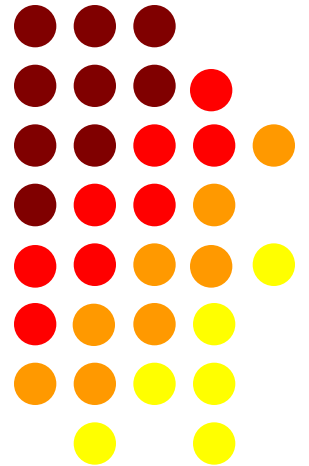
# Types of Software's

Basis	System Software	Application Software
<b>Basic</b>	System Software manages system resources and provides a platform for application software to run.	Application Software, when run, perform specific tasks, they are designed for.
<b>Language</b>	System Software is written in a low-level language, i.e. assembly language.	Application Software is written in a high-level language like Java, C++, .net, VB, etc.
<b>Run</b>	System Software starts running when the system is turned on, and runs till the system is shut down.	Application Software runs as and when the user requests.
<b>Purpose</b>	System Software is general-purpose.	Application Software is specific-purpose.
<b>Examples</b>	Operating system.	Microsoft Office, Photoshop, Animation Software, etc.



# Programming For Problem Solving

## Lecture 2



# Introduction to Operating System

- An Operating System (OS) is a system program which provides an interface between computer user and hardware. Some popular Operating Systems include UNIX, Linux, and Windows etc.
- **Resource Manager/Allocator:** An operating system is termed as resource manager or resource allocator, as its provides all necessary resources (Hardware, Software or Files) to application execution inside a computer system, like to play a song, OS allocates operational mouse, monitor, speaker, ram, hard disk, buses, processor etc. to application.

## Following are some of important functions of an operating System.

- Process Management
- Memory Management
- Device Management
- File Management
- Security Management



# Classification of Operating System

Multi Programming OS	Multi Tasking OS	Multi Threading OS
<ul style="list-style-type: none"> <li>In a multiprogramming system there are one or more programs loaded in main memory which are ready to execute.</li> <li>Only one program at a time is able to get the CPU for executing its instructions (i.e., there is at most one process running on the system) while all the others are waiting their turn.</li> <li>The main idea of multiprogramming is to maximize the use of CPU time.</li> </ul>	<ul style="list-style-type: none"> <li>Multitasking refers to having multiple (programs, processes, tasks, threads) running at the same time.</li> <li>This term is used in modern operating systems when multiple tasks share a common processing resource (e.g., CPU and Memory). At any time the CPU is executing one task only while other tasks waiting their turn.</li> </ul>	<ul style="list-style-type: none"> <li>Multithreading is an execution model that allows a single process to have multiple code segments (i.e., threads) run concurrently within the “context” of that process.</li> <li>Multiple threads of a single process can share the CPU in a single CPU system or (purely) run in parallel in a multiprocessing system</li> </ul>



# CUI Vs GUI Operating System

BASIS	CLI/CUI	GUI
<b>Basic</b>	Command line interface enables a user to communicate with the system through commands.	Graphical User interface permits a user to interact with the system by using graphics which includes images, icons, etc.
<b>Device used</b>	Keyboard	Mouse and keyboard
<b>Ease of Use</b>	Hard to perform an operation and require expertise.	Easy to perform tasks and does not require expertise.
<b>Memory</b>	Low memory consumption	High memory consumption
<b>Appearance</b>	Can't be changed	Custom changes can be employed
<b>Speed</b>	Fast execution speed	Slow execution speed



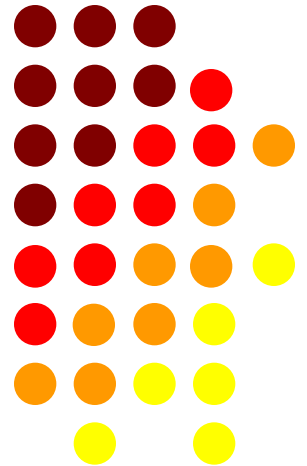
# Android Vs Windows Operating System

Basis	Android	Windows
<b>Graphics</b>	Relatively Optimized	Full fledged
<b>Source Code</b>	Open source code	Close source code
<b>Security</b>	Comparatively Less	Slightly More
<b>Application</b>	Ahead For Mobile	Known For PC
<b>Developer</b>	Google	Microsoft



# Programming For Problem Solving

## Lecture 3





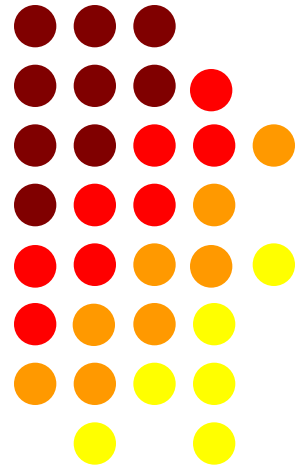
# Generation of Programming Languages

Low level language	High level language
<b>Advantage</b>	<b>Disadvantage</b>
They are faster than high level language.	They are comparatively slower.
Low level languages are memory efficient.	High level languages are not memory efficient.
No need of translator except assembler for AL.	Compiler & Interpreter is needed to convert HLL.
<b>Disadvantage</b>	<b>Advantage</b>
Low level languages are difficult to learn.	High level languages are easy to learn.
They are machine dependent and are not portable.	They are machine independent and portable.
They are more error prone.	They are less error prone.
Debugging and maintenance is difficult.	Debugging and maintenance is comparatively easier.
Ex: Machine & Assembly Language	Ex: C, C++, Java



# Programming For Problem Solving

## Lecture 4



# INTEGRATED DEVELOPMENT ENVIRONMENT

## Editor

- It is a tool of IDE much like a notepad that is used to write or edit the source code(a program written in C language) of any program.
- Editors are software programs that enable the user to create and edit text files.
- In the field of programming, the term editor usually refers to source code editors that include many special features for writing and editing code.

## Preprocessor

- It is a program that process the source code before it passes through The compiler and convert source code into expanded source code by mean of preprocessor directives.
- This is the first phase through which source code is passed. This phase include:
  - 1.**Removal of Comments**
  - 2.**Expansion of Macros**
  - 3.**Expansion of the included files.**
  - 4.**Conditional compilation**

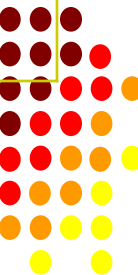


# Role of Assembler, Compiler & Interpreter

**Assembler:** An assembler is a system program, which convert an assembly language program in to machine language program, Example (Merlin, Vasm).

**Compiler:** While compiler & interpreter is a system program will convert high level language program in to machine language program.

Basis	Compiler	Interpreter
<b>Definition</b>	It Scans the entire program and translates it into machine code (As Whole)	Translates program one statement at a time (Line by Line)
<b>Error Display</b>	Display the syntax errors as a whole	Display the syntax errors line by line
<b>Execution Time</b>	The overall execution time is faster	The overall execution time is slower
<b>Debugging</b>	Debugging is hard	Debugging is easy
<b>.exe file</b>	. exe file is created using compiler	. exe file is not created using interpreter
<b>Example</b>	Programming language like C, C++ uses compilers. (Turbo C, GCC)	Programming language like Python, Ruby, Java uses interpreters. (Java Interpreter)

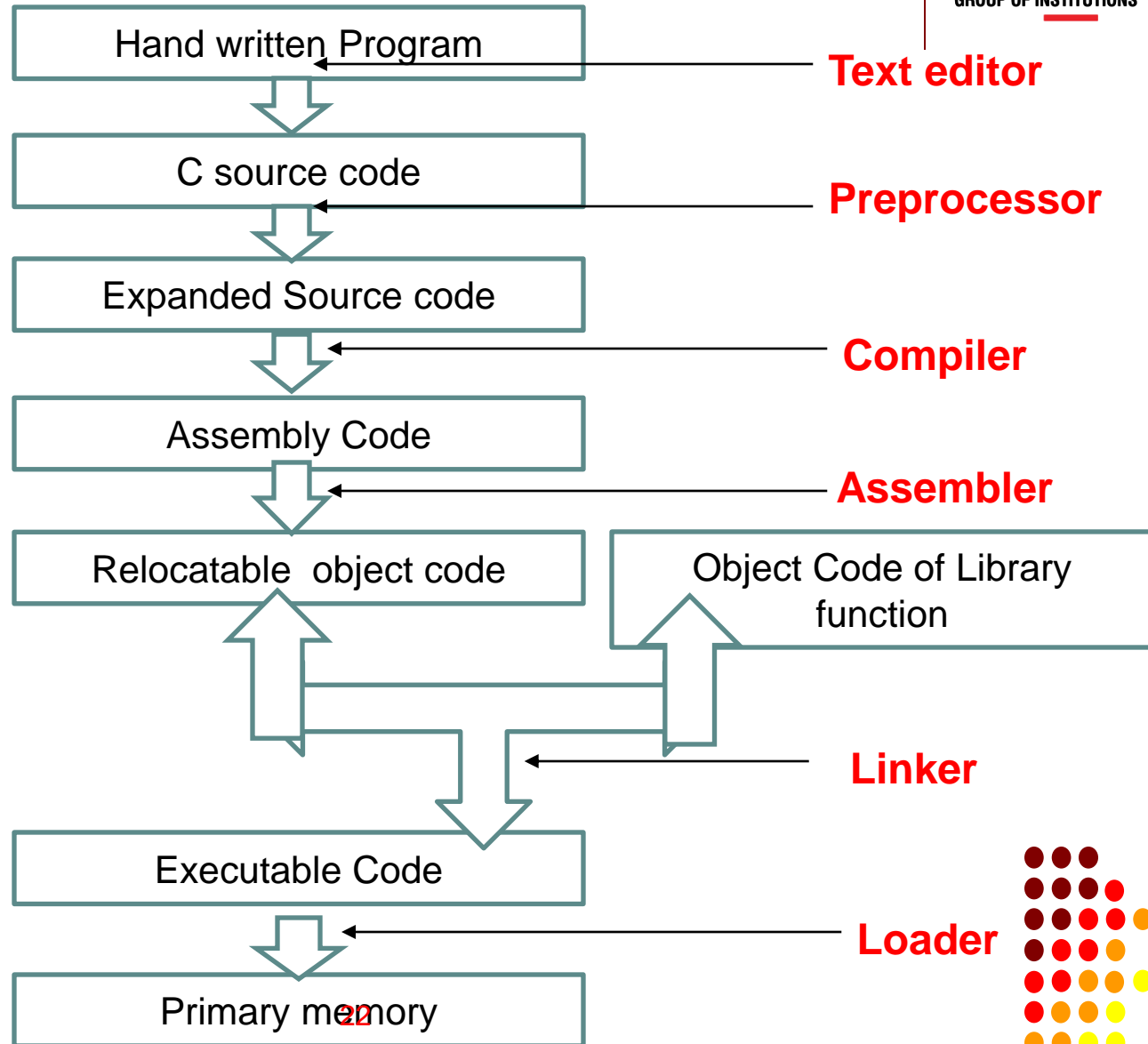


# Linker and Loader

- **Linker:**
- It is a program which combines various pieces of relocatable object code and data together to form a single executable code that can be loaded in primary memory.
- Linker is a program in a system which helps to link a object modules of program into a single object file.
- It takes object modules from assembler as input and forms an executable file as output for loader.
- Linking is performed at both compile time, when the source code is translated into machine code and load time, when the program is loaded into memory by the loader. Linking is performed at the last step in compiling a program.
- **Loader:**
- It is a system program or a part of an operating system that is responsible for loading the executable code into primary memory for its execution.
- Loader is the program of the operating system which loads the executable from the disk into the primary memory(RAM) for execution.
- It allocates the memory space to the executable module in main memory and then transfers control to the beginning instruction of the program.

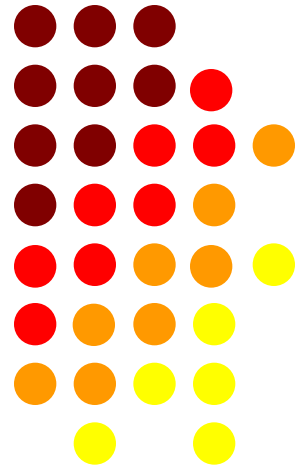


# Program Execution Cycle



# Programming For Problem Solving

## Lecture 5



# Algorithm & Its Characteristics

- A step-by-step method of solving a problem or making decision is termed as algorithm.

## Properties of the algorithm

- **Input.** An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins.
- **Output.** An algorithm has one or more outputs i.e., quantities which have a specified relation to the inputs.
- **Finiteness.** An algorithm must always terminate after a finite number of steps.
- **Definiteness.** Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case.
- **Effectiveness.** An algorithm is also generally expected to be effective. This means that all of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time.





# Algorithm & Its Characteristics (Cont..)

## Advantages of algorithm

- An algorithm uses a definite procedure which makes it easy to understand.
- It is not dependent on any programming language, so it is easy to understand.
- Every step in an algorithm has its own logical sequence so it is easy to debug.
- By using algorithm, the problem is broken down into smaller pieces or steps.

## Disadvantages of algorithm.

- Writing algorithm takes a long time & have no standard format.
- An Algorithm is not a computer program; it is rather a concept of how a program should be.



# Example:- How to make Tea ??

It requires number of steps to make perfect tea

- step:-1    Need pan
- step:-2    Pour water and tea leaf
- step:-3    Boil for 10 minutes
- step:-4    Add some sugar and milk
- step:-5    Boil for next 10 minutes
- step:-6    Filter it
- step:-7    Serve it

so we are performing task step by step.



# Flowchart & Its Notations

- Flowchart is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

## Advantages of flowchart:




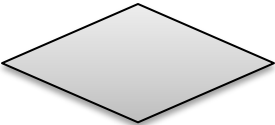
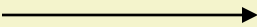
- The Flowchart is an excellent way of communicating the logic of a program.
- It is easy and efficient to analyze problem using flowchart.
- It helps the programmer to write the program code.

## Disadvantage of flowchart

- The flowchart can be complex when the logic of a program is quite complicated.
- Drawing flowchart is a time-consuming task.
- Difficult to alter the flowchart & uses special sets of symbols for every action.
- It is just a visualization of a program; it cannot function like an actual program.



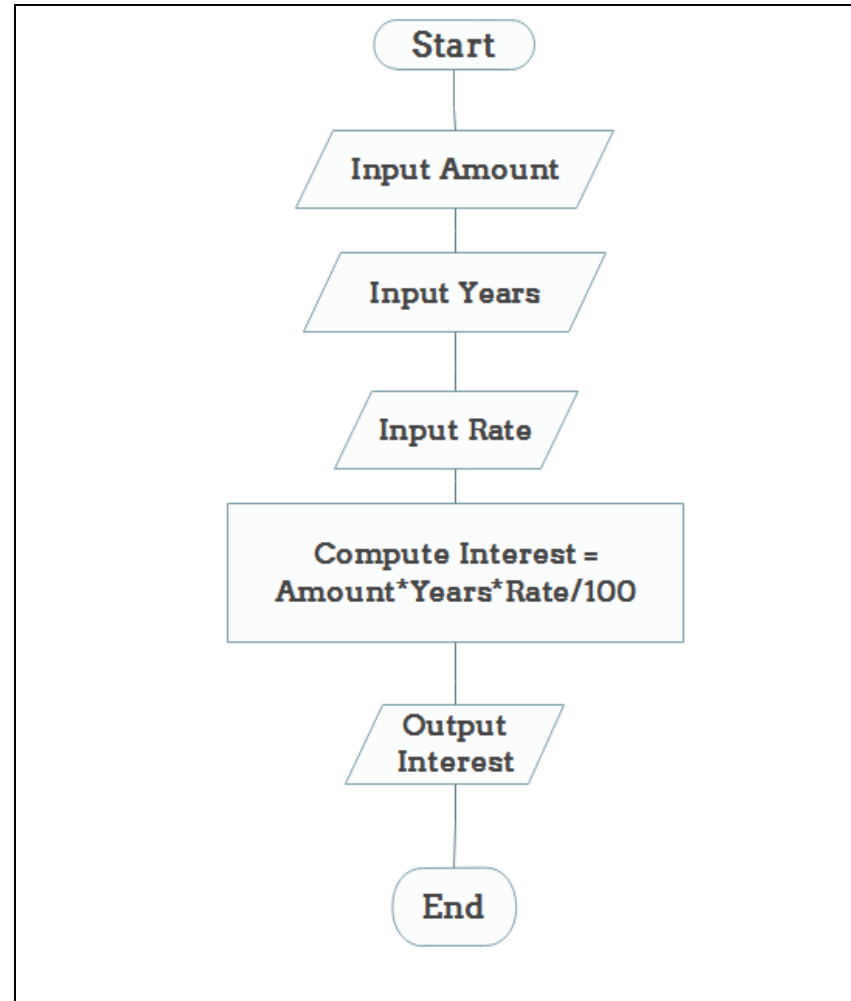
# Flowchart & Its Notations (Cont..)

Symbol	Symbol Name	Purpose
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.



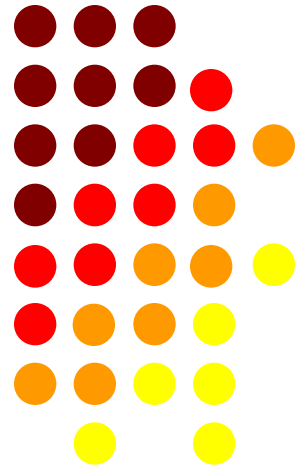
## Example

- To Draw the flow chart of simple interest.



# Programming For Problem Solving

## Lecture 6



# Concept of Pseudocode

- **Definition:** Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations.
- It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program's flow, but excludes underlying details.
- System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly.

## Advantages of pseudo code:

- Pseudocode is understood by the programmers of all types.
- It enables the programmer to concentrate only on the solution.

## Disadvantages of pseudo code:

- It cannot be compiled into an executable program.



# Concept of Pseudocode (Cont..)

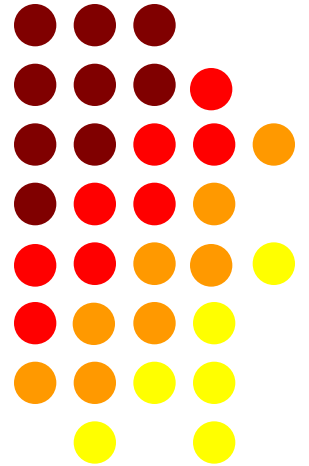
- **Example: Pseudocode to calculate the average marks of a class having 10 students**
- Set total to zero
- Set grade counter to one
- While grade counter is less than or equal to ten
- Input the student marks
- Add the marks into the total
- Set the class average to the total divided by ten
- Print the class average.





# Programming For Problem Solving

## Lecture 7



# STRUCTURE OF C PROGRAM

- **Documentation section** : It consists of a set of comment lines giving the name author date etc. of program and other details.
- **Link section**: It provides information or instructions to the compiler to link functions from the system library.
- **Definition section**: It defines all symbolic constants.
- **Global declaration section**: Variables that are declared outside all the functions and are used in more than one function.
- **Main function section**: This section contains two parts declaration part and execution part.
- **Subprogram section**: It contains all the user defined functions that are called in main functions.



# First C Program

- Write a program in C to find the addition of two number.

```
#include <stdio.h>
int main()
{
    int number1, number2, sum;
    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);
    sum = number1 + number2;
    printf("\nThe Sum is%d",sum);
    return 0;
}
```

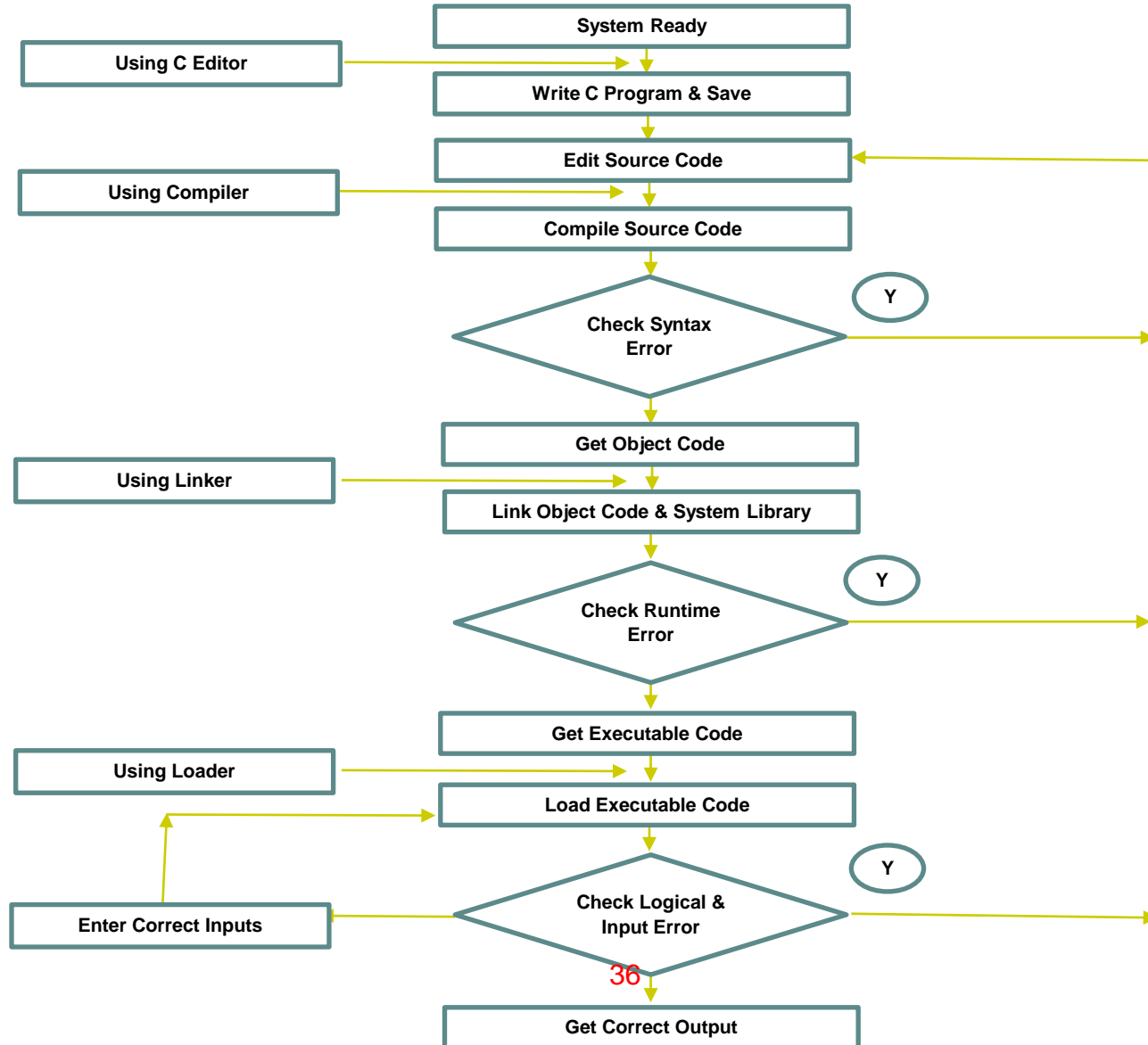
**Output:-**

**Enter two number 5 6**

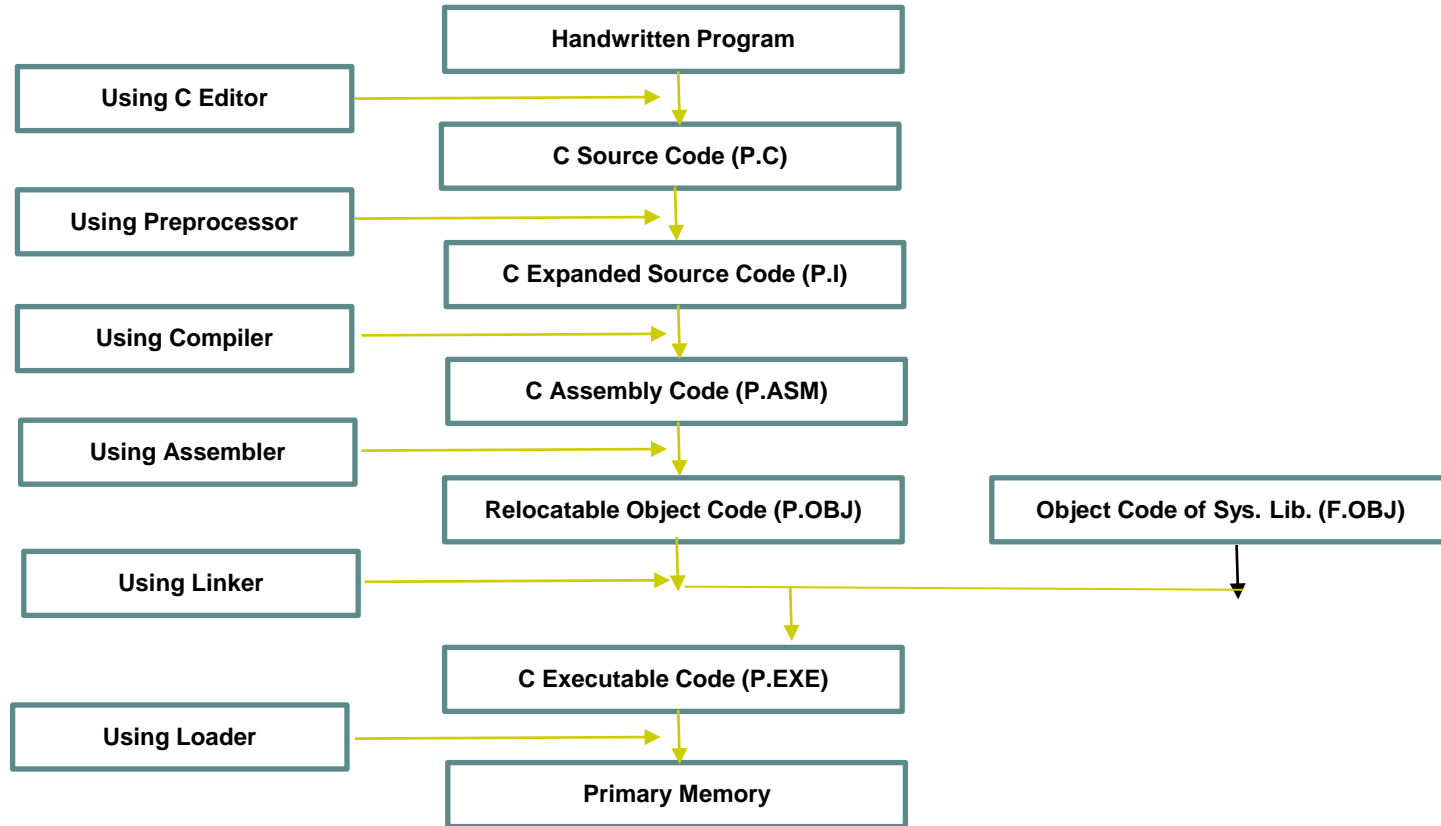
**The Sum is 11**



# Compilation & Execution Process



# File Based Structure of C



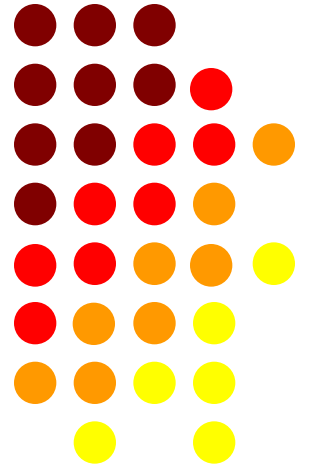
# Types of Programming Errors

- **Syntax Error:** The errors which arises due to violation of any rules of C language, during the development of a program, those errors are known as syntax error. These errors are identified by compiler during compilation. Example missing of semicolon.
- **Runtime Error:** The error which occurs during the execution of program, those errors are known as runtime error. These error are identified by linker during linking object code with system library. Example using print instead of printf.
- **Logical Error:** The errors which arises due to usage of wrong expression, formula for logic in program, those errors are called as logical error. Example using  $2 * 3.14 * r$  as area of circle. These errors are not identify by compiler or linker.
- **Input Data Error:** The error which occur during the data entry process, because of entering the wrong input values, during the execution of a program are called as input data error.



# Programming For Problem Solving

## Lecture 8



# Input Output Statement in C

- The input output statement are classified into two categories:

Type	Input Function	Output Function
Formatted I/O	scanf()	printf()
Unformatted I/O	getchar(), getch(), gets(), getche()	puts() , putch(), putchar()





# Input Output Statement in C

**scanf() Function** : The function used to get input from the user during execution of the program and stored in a variable of specified form is called scanf() function.

## Syntax:

**scanf("format string",& variable name);**

## Example:

Single Input Example: `scanf("%d",&a);`

Multiple Input Example: `scanf("%d%d", &a,&b);`



# Input Output Statement in C

**printf() Function:** The function used to display text, constant or value of variable on screen in specified format is called printf() function.

## Syntax:

```
printf("format string", argument list);
```

## Example:

```
printf("hello world");           // printf() with no argument list
```

```
printf("Value=%d",a);           //printf() with one argument
```



# Input Output Statement in C

- **getchar():** The getchar function is a part of the standard C input/output library. It returns a single character from a standard input device (typically a keyboard). The function does not require any arguments, though a pair of empty parentheses must follow the word getchar.

## Syntax

- **character variable = getchar( );**
- Where character variable refers to some previously declared character variable
- **putchar():** The putchar function like getchar is a part of the standard C input/output library. It transmits a single character to the standard output device (the computer screen).

## Syntax

- **putchar(character variable)**
- Where character variable refers to some previously declared character variable.



# Input Output Statement in C

<b>getch()</b>	<b>getche()</b>
getch() is used to get a character from console but does not echo to the screen.	getche() is used to get a character from console, and echoes to the screen.
It reads a single character directly from the keyboard, without echoing to the screen.	getche() reads a single character from the keyboard and echoes it to the current text window.



# Role of Escape Sequence in C

- Backslash character constant is used to format output on time of execution.

Character	Escape Sequence	Result
Bell	\a	Beep Sound
Back Space	\b	Moves Previous Position
Horizontal Tab	\t	Moves next horizontal tab
Vertical Tab	\v	Moves next vertical tab
Newline (line feed)	\n	Moves next line
Form feed	\f	Moves initial position next page
Carriage return	\r	Moves beginning of the next line
Single quote	\'	Present Apostrophe mark
Double quotes	\"	Present double quotes
Backslash	\\	Present Back slash mark



# Role of Escape Sequence in C

- Backslash character constant is used to format output on time of execution

## Example:

- `main()`
- `{`
- `printf("\nab");`
- `printf("\bsi");`
- `printf("\rha");`
- `}`

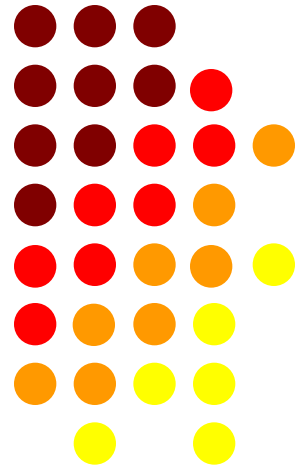
## Stepwise Output

- `—`
- `a b`
- `a s i`
- `h a i`



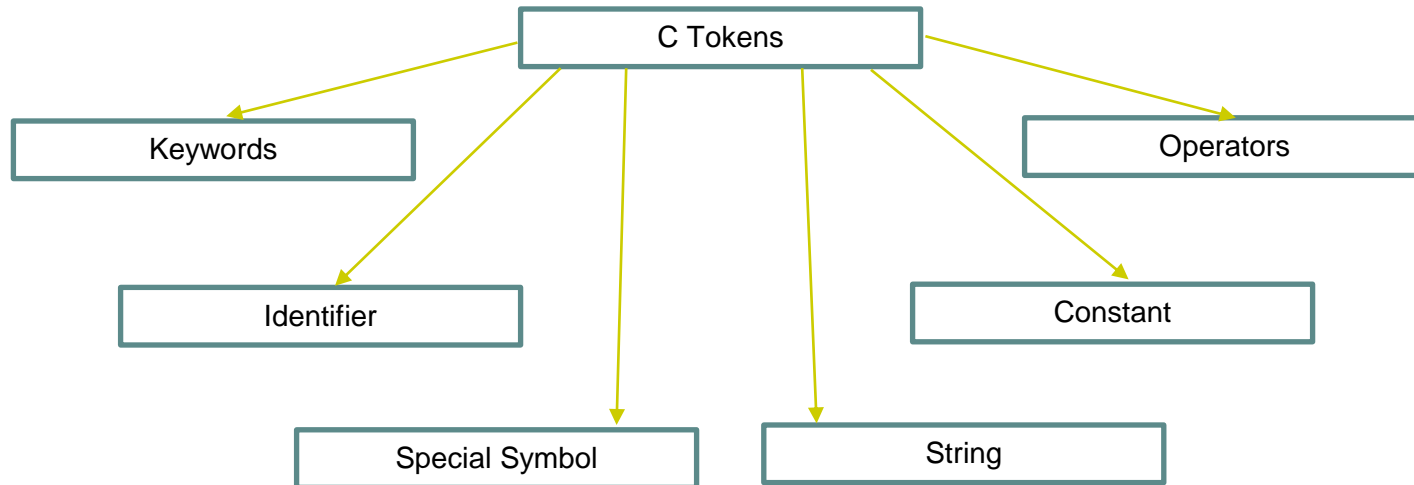
# Programming For Problem Solving

## Lecture 9



# Introduction to C Tokens

- The basic fundamental units used in C language are known as C tokens. C has six tokens as given below:





# Introduction to C Tokens (Cont..)

- **Keyword** are also known as reserved words, whose meaning is already defined in C library. There are mainly 32 keywords some are : int, char, float, double, if , else, break, continue ,case, void etc.
- **Identifier** are also known as variable, these are those entity whose value may be changed any time during execution of program. It is very important to declare the type of variable before it is used in program.

## Rules For Naming Variables

- A variable name is any combination of alphabets, digits or underscore
- The first character must be always alphabet or underscore
- No comma or blank space is allowed within variable name
- Only underscore is allowed in special symbols
- Examples are ac ,\_cc ,c\_a



# Introduction to C Tokens (Cont..)

- **Constant** are entity or value that does not change during the execution of programs like 3.14

## Rules for character constant

- It is a single alphanumeric digit.
- The character is enclosed in single quotes.
- The maximum length of a character constant is one.
- Example 'a', '1', '@'
- **Variable** refers to the name of the memory location whose value can be change during program execution.

**Eg:-** int a;

- **String** is the collection of characters, digits or special symbols enclosed in double quotes, terminated by null character like "Hello"



# Introduction to C Tokens (Cont..)

- **Example: Identify C tokens in the code given below**

- void main()
- {
- int r;
- float ac, cc;
- clrscr();
- printf("enter radius");
- scanf("%d",&r);
- ac=3.14\*r\*r;
- cc=2\*3.14\*r;
- printf("area=%f\tcircum=%f",ac,cc);
- getch();
- }

S. No	C Tokens	Example
1	<b>Keywords</b>	void, int, float
2	<b>Identifiers</b>	r, ac, cc, printf, scanf
3	<b>Constant</b>	3.14, 2, \t
4	<b>Operators</b>	=, *
5	<b>String</b>	"enter radius"
6	<b>Special Symbol</b>	{, }, ;



# Overview of C Datatypes

- Datatype is used to declare the variable. It determine the type of value and the range of values that can be stored inside a variable.

Datatype	Format specifier	Size	Range
char	%c	1 byte	-128 to 127
signed int	%d	2 bytes	-32768 to 32767
signed short int	%d	2 bytes	-128 to 127
signed long int	%ld	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned int	%u	2 bytes	0 to 65,535
unsigned short int	%u	2 bytes	0 to 255
unsigned long int	%lu	4 bytes	0 to 4,294,967,295
float	%f	4 bytes	3.4E -38 to 3.4E +38
double	%lf	8 bytes	1.7E -308 to 1.7E +308
long double	%Lf	10 bytes	3.4E -4932 to 1.1E+4932

