C Programming

Ch. 001

Introduction of C Language

Language

- What is a Language?
  - Language is a collection of word and symbols.
  - Language is used to perform certain task of activities and to establish a communication between people.

Language

- Language is generally used to build up a communication between persons, machines or computers.
- Generally human beings used different languages in different region for effective communication.
- Languages is used to give some instructions / information / commands / details to other person.

Language

- Basically, computer is a machine and machine cannot do any thing without instruction/command.
- If we want to work with computer and wish to do some work as per our requirement then we must give some instructions / command such language.
- Generally computer understand some special languages only. These languages are known as **computer languages.**

Computer Languages

- There are mainly three different languages...
  - Low Level Languages
  - High Level Languages
  - Middle Level Languages

Low Level Programming Language

- Low level programming language is also known as Binary Language too.
- Binary Language consists only two characters **0 [Zero]** and **1 [One]**.
- Low level programming language does not require translator like compiler or interpreters.
- Machine can direct understand Low level programming with fast execution speed.

Low Level Programming Language

- We can divide low level programming language in two types...

A. First Generation Language (1GL)

- Or Machine Language

B. Second Generation Language (2GL)

- Or Assembly Language

A. 1GL - Machine Language

- 1GL is also known as
  - Binary Language / Machine Language
- Computer can only understand machine level language which is known binary language.
- This is only the language that machine can process directly, without any prior translation.
- For Example,

  0011 1100

A. 1GL - Machine Language

- Advantage of Binary / Machine Language
  - Program execution speed is fast.
  - This program will uses less memory.
  - No translator required because processor can directly understand machine instructions.

A. 1GL - Machine Language

- Disadvantage of Binary / Machine Language
  - Difficult to use & remember :
    - It is very difficult to remember codes for a programmer because the codes are written in BINARY LANGUAGE.
    - If programmer will place a single mistake then it will generate different meaning output.

A. 1GL - Machine Language

- Disadvantage of Binary / Machine Language
  - Machine dependent language :
    - Before writing program in binary language, the programmer has to know machine architecture because this language is machine dependent language.

A. 1GL - Machine Language

- Disadvantage of Binary / Machine Language
  - Difficulty in Error:
    - It is hard to understand and remember the various combinations of 1's & 0's representing.
    - This make it difficult for a programmer to develop error free program.
  - Difficult debugging process :
    - Checking machine instructions to find out errors are as tuff as programming.

A. 2GL - Assembly Language

- In the first generation language, programmer has to write codes using 1 and 0's bit format.
- In the second generation language programmer can use symbols in place of bits.
- In 2GL, such types of symbolic representations are also known as mnemonic codes, these codes are used in place of binary codes.
- A program is written with mnemonic codes is known as assembly language program.

A. 2GL - Assembly Language

- Example :

LD A, 5 = Load register A with 5

LD B, 4 = Load register B with 4

ADD A,B = Add value of register B in to value of register A.

A. 2GL - Assembly Language

- Advantages of Assembly Language :
  - It is easy then machine language.
  - Assembly code is more readable and under stable because it is written with the help of symbolic words or predefine words.
- Disadvantages of Assembly Language :
  - It is also machine dependent language.
  - Machine dependent means a program is designed for a processor will not work on a different type of processor.

A. 2GL - Assembly Language

- Disadvantages of Assembly Language :
  - To write a program in assembly is again difficult and time consuming.
  - Program execution speed is slow as compare to machine language.

3GL - High Level Programming Language

- High level languages are user friendly language because their instructions are similar to human languages.
- High level languages have a set of grammar that makes it easy for a programmer to write programs and identify the correct errors in a program.
- All high level languages have its own compiler or interpreter to convert its code into machine code (Binary Codes).
- For Examples,
  - Cobol, Fortran, Basic, Java etc.

3GL - High Level Programming Language

- Advantages of high level programming languages…
  - Readability of program :
    - Low (Machine) level languages are not in readable but High level languages are easy read.
  - Program Portability (Machine Independent):
    - It is a machine independent.
    - The code of high level languages can be run on different machines easily with little or no change in codes.

3GL - High Level Programming Language

- Advantages of high level programming languages…
  - Easy to debug a program:
    - In high level programming language it is easy to find out errors form the code and we can easily remove the errors…
  - Fast program development :
    - The statements of these programming languages are more similar to English language so program development will be fast.

3GL - High Level Programming Language

- Disadvantages of high level programming languages…
  - Program development is fast but program execution speed is slow as compare to low level programming language.
  - Must have a compiler or interpreter because High level programming language codes must require to be translate in to machine codes fist.

Middle Level Programming Language

- Middle Level Programming is a language which has mixed power of Low Level Programming Language and High Level Programming Language.
- Middle Level Programming is Easy for Programming compare to Low Level Programming.
- Middle Level Programming is Fast for execution compare to High Level Programming.
- Middle level programming reduces the negative point and add some advanced points.

POP

- What is POP ?
  - POP is a procedure oriented programming language.
- List of POP
  - C
  - Basic
  - Fortran are example of POP.

OOP

- What is OOP ?
  - OOP is a Object oriented programming language.
- List of OOP
  - C++
  - C#
  - Java are example of OOP.

What is translation ?

- Translation means to convert source-language by target-language.
- Examples
  - Gujarati **To** Hindi
  - Gujarati **To** English
  - English **To** Gujarati
  - Text **To** Binary
  - Binary **To** Text

What is language translator ?

- As we know that computer understands only the instruction written in the Machine language.
- Therefore a program written in any other language should be translated to Machine language.
- For these purpose special programs are available they are called **translators or language processors**.

Translator :

- Translator is a special program. This translator accepts the user program and checks each statement and produces a set of Machine language instructions.
- There are tow types of Translators.
  - Complier
  - Interpreters

Translator :

- Compiler:
  - A Compiler checks all the entire program written by user.
  - If program is free from error and mistakes then compiler produces this program as a complete program.
  - If program founds some errors in the program then it does not execute a single statement of the program.
  - So Complier translate whole program in Machine language before it starts execution.

Translator :

- Interpreters
  - Interpreters perform the similar job like complier but in different way.
  - Interpreters translates one statement at a time and if it is error-free then executes that statement.
  - This continues till the last statement in the program has been translated and executed.

Translator :

- Interpreters
  - Thus Interpreter translates and executes the statement before it goes to next statement. When it found some error in statement it will immediately stop the execution of the program.
  - Since Compiler or Interpreter can translate only a particular language for which it is designed one has to use different compiler for different languages.

Translator :

- Difference between Compiler & Interpreter

Compiler	Interpreter
Compiler first checks all the statement for error and provide lists of all the errors in the program.	Error finding is much easier in Interpreter because it checks and executes each statement at a time.
Compiler take less time then interpreter because it translate and executes all statement at same time.	Interpreter take more time for the execution of a program compared to Compilers because it translates and executes each statement one by one.

Introduction of Programming Concept

- The programming concept is used to make programming easy.
- To perform various program we can follow two method.
  - Algorithm
    - To represent solution of problem in written steps we can use a method that is known as algorithm.
  - Flowchart
    - To represent solution of program in graphical steps we can use a method that is known as flowchart.

Introduction to C : History Of C

- C programming was developed using different way of development.
- It was developed by step as given below.

Language	Year	Developed by
ALGOL (ALGOrithmic Language)	1960	International Committee
CPL (Combined Programming Language)	1963	Cambridge University

History Of C

Language	Year	Developed by
BPCL (Basic Combined Programming Language)	1967	Martin Richards & Cambridge University
B (B Language)	1970	Ken Thomson at AT&T Laboratory
Traditional C (C Language)	1972	Denis Ritchie at AT&T Laboratories

History Of C

Language	Year	Developed by
ANSI C (American National Standards Institute)	1989	*ANSI Committee*
C99	1999	Standardization Committee

What is C ?

- C is programming language.
- It is also known as middle level programming language.

C as a Middle Level Programming Language

- C Programming is faster then Low Level Languages.
- Program execution speed is faster then High Level Programming Languages.
- C behaves as High Level Language through functions and reusability.
- C language gives access to the low level memory through pointers.
- Because features of Low Level Programming Language and High Level Programming language **C is known as Middle Programming Language**.

Why C ? or Importance of C

- C programming codes are very efficient and fast for programmer and developer.
- C programming has variety of data types and powerful operators.
- C programming is faster than other programming languages.
- C is a powerful and flexible language which helps system developers and programmer to deliver various complex tasks easily.
- C programming can complete projects like, operating system, word processors, graphics, spreadsheets and even compilers for other languages.

Why C ? or Importance of C

- C program written for one computer system can be run on another system with little or no modification.
- C program has its own built in library functions.
- Write C program with UDF (User Defined Function) makes program more simple and easy to understand.
- These all points makes C programming more important.

Difference Between **Traditional and Modern C**

Topics	Traditional C	Modern C
Development	Early 1972	In year 1999. So, it is also known as C99
Developer	Dennis Ritchie	Standardization Committee
Speed	Very Low	Very high speed
Features	Few features are exists	Many new features are added
Keywords	Only 32 keywords	37 keyword in modern C

How To Start Turbo C ?

We can start C using following methods...

1. Start

   Programs > Turbo C++ IDE

2. Start

   Run... > C:\TC\TC.EXE

3. Short Cut

First Screen Of C programming

- In the first screen we can see following titles...

Title bar

Menu bar

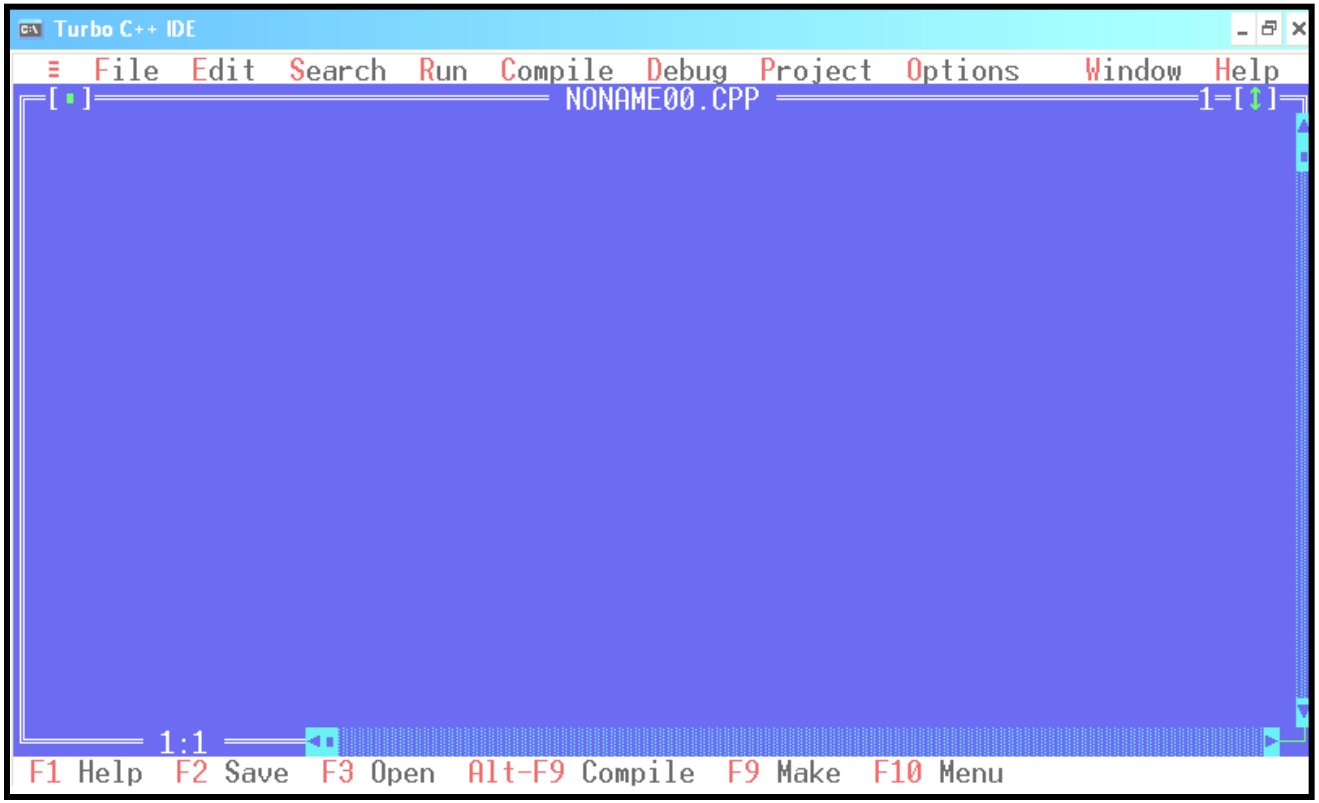
File Name

Scroll bar

Status bar

ShortKey Bar

Etc.



Shortcut Used in C editor

What to DO ?	Shortcut
Save a program file	F2
Open a saved file	F3
To maximize editing window	F5
Undo last changes in Editor	Alt+BackSpace
Redo last changes by Undo	Shift+Alt+B.Space
Copy selected text	Ctrl + Insert Key
Cut selected text	Shift + Del
Paste copied/cut text	Shift + Insert Key

Shortcut Used in C editor

What to DO ?	Shortcut
Clear Selected Text	Ctrl + Del
Compile a C Program	Alt + F9
Make a C Program	F9
Run a C Program	Ctrl + F9
Step by Step Debugging	F7 or F8

Features of C language

- There are several reasons why many computer professionals feel that C is at the top of the list:
  - Flexibility
  - Portability
  - Compactness
  - Reusability

Features of C language

- Flexibility:
  - C is a powerful and flexible language.
  - C is a popular language preferred by professional programmers.
  - As a result of flexibility, a wide variety of C compilers and helpful accessories are available.

Features of C language

- Portability:
  - C is a portable language. Portable means that a C program written for one computer system can be run on another system with little or no modification.
- Compactness:
  - C is a language of few words, containing only a handful terms, called keywords, which serve as the base on which the language's functionality is built.

Features of C language

- Reusability:
  - C is modular, C code should be written in routine called functions.
  - These functions can be reused in other applications or programs.
  - By passing pieces of information to the function, you can create useful, reusable code.

Structure of C Program

Documentation Section

File include section (Link Section)

Define Symbolic Constant

Global declaration Section

Function Prototype

main( ) (Main Function Section)

{ declaration part

executable part

}

User defined Function or Sub program

Structure of C Program

- Documentation Section:
  - This section contains set of comment line consist of details like program name, author name and purpose of functionality of the program.

```
//          For single line comment
Example :     // Program 1
/*   */   For multi line comment
Example:      /* Program 1
              My First Program of C
              Print Name of TEAM   */
```

Structure of C Program

- File Include Section or Link Section:
  - This section consists of instructions to be given to the compiler to link functions from the system library.
  - As example if you want to use some mathematical function then you have to define link for math.h file in link section.
  - For Example,

```
#include<stdio.h>
#include<math.h>
```

Structure of C Program

- Define Symbolic Constant
  - This section defines all the symbolic constants and function prototype.
  - For example

    PI=3.14.
  - By defining symbolic constant one can use these symbolic constant instead of value.

    #define PI 3.14

    #define Temp 35

    #define P printf

Structure of C Program

- Function Prototype
  - This section defines user define function as a prototype.
  - Example

```
void star( );
int sum(int, int);
```

Structure of C Program

- Global Declaration Section:
  - This section contains declaration of variables which are used by more then one function in a program.
  - Generally this kind of declaration will be held at the top of all function.

Structure of C Program

- Main Function Section:
  - Every C program must have a main( ) this section contains two parts
    - 1.Declaration Part
      - It declares all the variables used in executable part.
    - 2.Executable Part
      - Here at least one statement in the executable part to execute program.

Structure of C Program

- User Define Function or Sub program :
  - This kind of functions are defined outside of main function.
  - This function can be called from any point or anywhere from the program.
  - Generally these functions are defined to distribute the program in to small part.

C Character Set

- The characters which can be used in a program are known as Character Set.
- C character set can list as given.

1. Letters
2. Digits
3. Special Symbol
4. White Spaces

C Character Set

- 1. Letters (Alphabets)
  - Upper Case **A** to **Z**
  - Lower Case **a** to **z**
- 2. Digits
  - **0** to **9** with decimals

C Character Set

- 3. Special Symbols

		(_)	Underscore
,	Comma	&	Ampersand
.	Dot	^	Caret sign
;	Semicolon	*	Asterisk
:	Colon	-	Minus Sign
?	Question Mark	+	Plus Sign
‘	Single Quote	<	Opening Angular (Less than sign)
“	Double Quote	>	Closing Angular (Greater than sign)

C Character Set

!	Exclamation Mark	(	Left Parenthesis
|	Vertical Bar(Pipe)	)	Right Parenthesis
/	Slash	[	Left Bracket
\	Backslash	]	Right Bracket
~	Tiled	{	Left Brace
%	Percentage Sign	}	Right Brace
$	Dollar Sign	#	NumberSign(Hash)

C Character Set

- 4. White Space
  - Blank Space
  - Horizontal Tab
  - Carriage Return
  - New Line
  - Form Feed

C tokens

- C language programs are made of various
  - Identifiers
  - Keywords
  - Constants
  - Operators
  - Strings and special symbols, these are called as **C tokens.**
- Any C language program can not be created without help of one or more tokens.

C tokens

- Identifiers
  - Identifier is a name of variable, constant, structure, union, arrays or function given by the programmer.

C tokens

- Keywords
  - Keyword is a word which is having a specific meaning for a particular language.
  - Keywords are reserved word by programming language.
  - Keywords are not used as identifiers.
  - There are 32 keywords available in C.

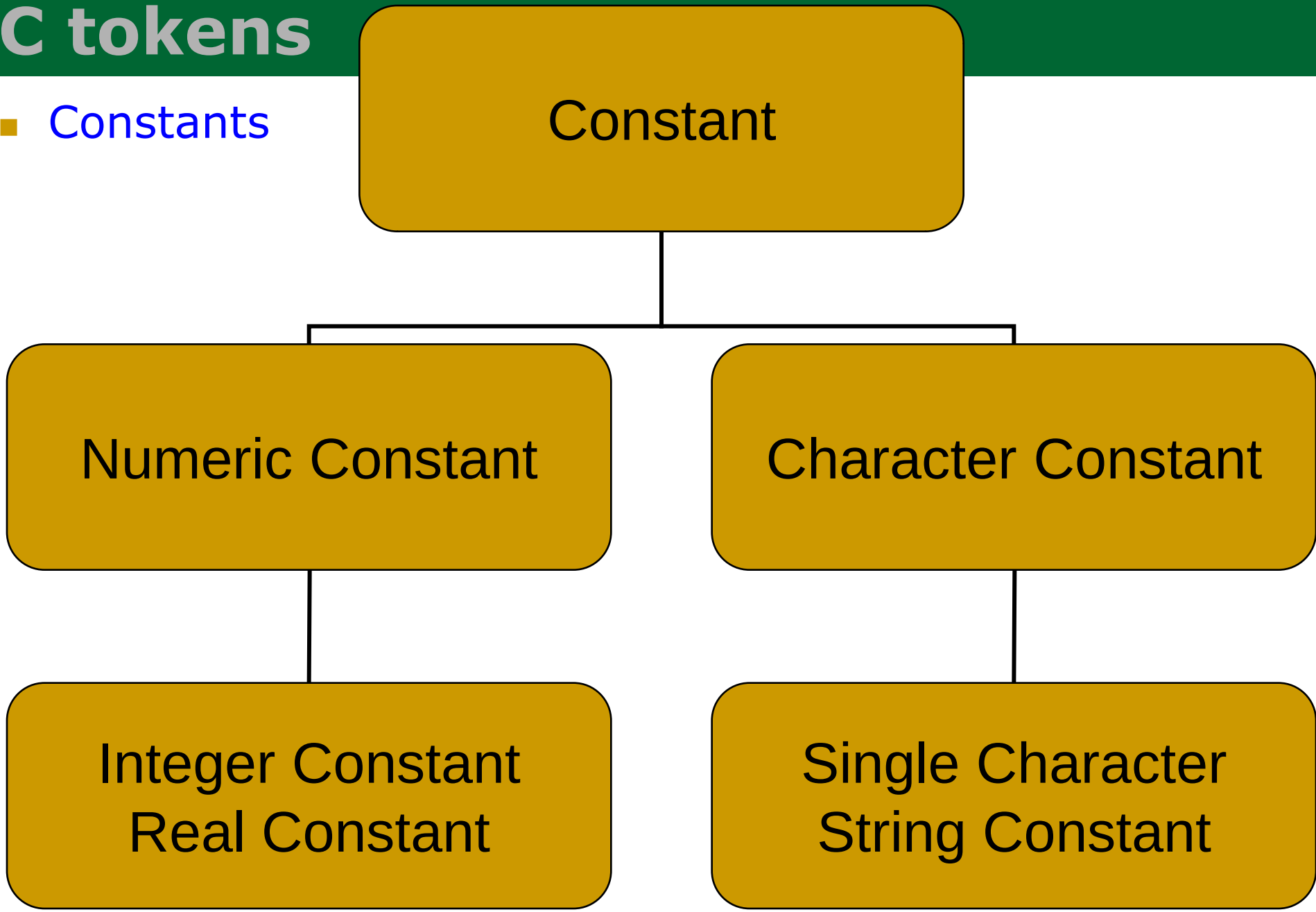
C tokens

- 32 Keywords

auto	break	case	char
const	continue	do	Double
default	else	enum	extern
float	for	goto	if
int	long	register	return
struct	static	short	signed
switch	sizeof	typedef	union
unsigned	void	volatile	while

C tokens

- Constants



C tokens : Numeric Constant

- Integer Constants
  - An integer constant is a combination of digits without decimal point.
  - An integer constant may be a positive or negative number.
  - Example,

    135, 0, -50, +54

C tokens : Numeric Constant

- Real / Float Constant
  - A real constant is a combination of digits with decimal point.
  - Examples,
    0.50, -12.11, 155.20

C tokens : Character Constant

- Character Constant (Single Character)
  - A single character constant is a single alphabet.
  - This character is enclosed with single quotation mark.
  - Examples,

    gender='M'

    status='Y'

C tokens : Character Constant

- Character Constant (Character String)
  - String constant is a group of characters with double quotation mark.
  - Examples,

    “hello”

    “WelCome”

    “365430”

C tokens

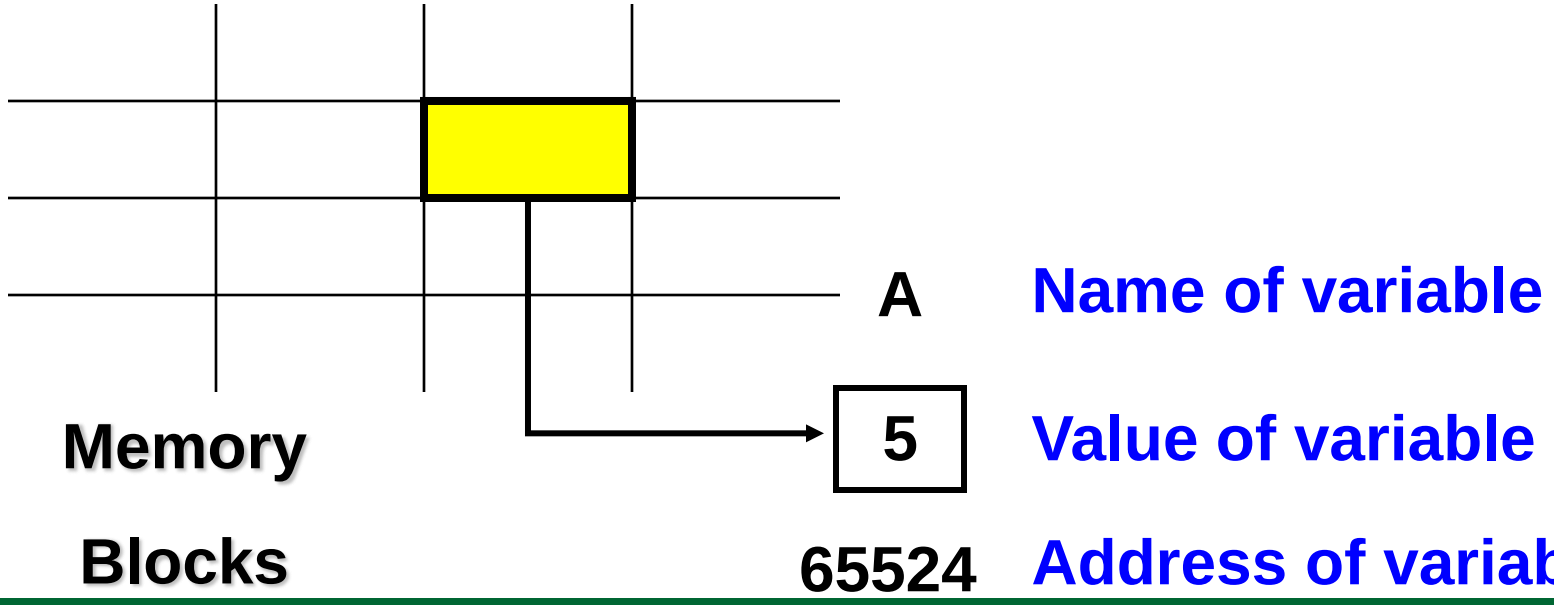
- Operators
  - Operators are the special symbols that report the computer that which type of arithmetic or logical operation to be performed on variable or operands.

Variables

- Computer has memory cells same as human mind to remembering the values.
- Each memory cells in the computer has unique address which will be in binary format.
- It is not possible to remember the address of each memory cell so that cell is introduced with a name which is called "**variable".**
- Variable is a storage area where the values can be stored. These values may be a fixed value or the value is inputted by the user.

Variables

- All the **C** variables have three attributes:
  - Name
  - Value
  - Address of memory location, where variable is stored
    - For Example



Rules for variable naming :

- The first character in identifier name must be an alphabet or underscore
- No commas or blanks are allowed within an identifier name.
- No special symbol other then an underscore can be used as in identifier name.
- An identifier name is any combination of 1 to 31 alphabets, digits or underscores without space.
- Do not create unnecessarily long variable names as it adds to you typing efforts.
- C language is case sensitive. Upper and Lower case both are allowed in C programming. But they are not equivalent. Here **NUM** and **num** both are different.

- What is Operator?
  - The symbols which are used to perform mathematical or logical operation those symbols are known as operator.
  - Like,
    - +, -, *, /, &&, ||, !=, <, >, etc.
- What is Operand?
  - Operand means the variable or value which is used before and after the operator.
  - Like
    - A+B (Here, **A** and **B** are operand and **+** is operator)

Operators

(1) Arithmetic Operator

(2) Relational Operator

(3) Logical Operator

(4) Assignment Operator

(5) Increment / Decrement Operator

(6) Conditional Operator

(7) Bitwise Operator

(8) Special Operator

Arithmetic Operator

- To make arithmetic processing we can use arithmetic operators.

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Reminder after integer division

Relational Operator

- To check relationship between two variable we can use different relational operators.

Operator	Meaning
<	Less then
<=	Less then or Equal to
>	Greater then
>=	Greater then or Equal to
= =	Equal to
!=	Not Equal to

Relational Operator

- For Example,

  a=1, b=2, c=3;

Expression	Interpretation	Logical Value
a<b	True	1
(a+b)>c	False	0
(a+b)>=c	True	1
(a+b)=c	True	1
(a+b)<c	False	0
(a+b)<=c	True	1
c!=3	False	0
c==3	True	1

Logical Operator

- To provide such logical operations in C we can use logical operators. As given below :

Operator	Meaning
&&	AND
||	OR
!	NOR

Logical Operator

- **Logic for && :** While all the conditions will be true using && then logical output will be true else output will be false.

Truth Table for &&

Input A		Input B	Output
True	&&	False	False
False	&&	True	False
False	&&	False	False
True	&&	True	True

Logical Operator

- **Logic for || :** While any of the condition will be true using || then logical output will be true else output will be false.

Truth Table for ||

Input A		Input B	Output
True	||	False	True
False	||	True	True
True	||	True	True
False	||	False	False

Logical Operator

- **Logic for ! :** While any of the condition will be true using || then logical output will be false else output will be true.

Truth Table for !

Input A	Output
True	False
False	True

Conditional Operator

- C programming offers the conditional operator used in C language is (?) question mark and (:) column.
- It is used for checking the condition and gives output according to the condition.
- If condition is true then it will execute the first value or it will execute the second value given in the syntax.

Conditional Operator

- Syntax :

  <condition>?<True Value>:<False Value>

- Purpose:
  - If condition will be true then it will return the true value else it will return the false value.

Conditional Operator

- Example:

```
a=10;

b=12;

c=(a>b)?a:b  // It will store value of B

printf("Value of C is %d",c);

sub1=23, sub2=40;

printf("Highest=%d",(sub1>sub2)?sub1:sub2);
```

Assignment Operator

- Assignment operators are use to assign any value to any data to any identifier. The most commonly sign used as assignment operator is = (equal).
- Identifier = Expression

```
a=10                    a=a-1
a=a-10                  a-=1
a= (a*2) +a             a=a*a
a=a+1                   a*=a
a+=1                    a=a/2    or   a/=2
```

Sizeof Operator

- To check the size of variable in bytes we can use sizeof operator.
- For Example,

```
int a=150;

float b=25000;

printf("Size of A :%d",sizeof(a));

printf("Size of B :%d",sizeof(b));
```

Increment/Decrement Operator

- C programming offers two special operator ++ and -- called as increment and decrement operators.
- There are “unary” operators are only able to work with proper variable. It will not work on constant or numeric value.
- Here a++ is valid but 6++ is invalid. These operators can be used either before or after their operand. Like a++ is valid and ++a is also valid.

Increment/Decrement Operator

- For Example,

```
a=10;
a++;     // now a =11
- -a;    // now a=10
++a;     // now a=11
a- -;    // now a=10
```

Bitwise Operator

- Bitwise operators are used to perform bit manipulations.

  Generally these operators used to perform hardware related task by c programming.

Operators	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift Left
>>	Shift Right

Bitwise Operator

- Example :

```
int a=7, b=5;
a & b    : 111 & 101 = 101 = 5
a | b    : 111 | 101 = 111 = 7
a ^ b    : 111 | 101 = 010 = 2
```

Special Operator

- C language supports many special operators too. They are used for some special processes in C programming. T
- he special operators used in C programming language are,
  - Comma (,)
  - Pointer Operators (& and *)
  - Member Selection (. and ->)
  - sizeof( ) operator

Hierarchy of Operators

1. First evaluate parentheses expression.
2. If parenthesized are nested, the evaluation begins with the innermost sub-expression.
3. Arithmetic operators without parentheses will be evaluated from left to right using the rules of operators.
4. There are two distinct priority levels of arithmetic operators.

   High priority : * / %

   Low priority : + -

Data Type of C Language

- We can use Integer In any variable we can assign different value.
- Before assign any value in any variable we must specify type of data.
- These data are distributed under different category and that is called data type as given.

1. int (Integer)
2. float
3. char (character)
4. double
5. void

C Data Type Introduction:

- Integer
  - This data type is used to store numbers without decimals.
  - Range between -32768 to +32767
  - Occupies 2 byte (16 bits)
  - Format code is %d
- Example,

```
int a,b,c;
a=10;
b=20;
c=a+b;
```

C Data Type Introduction:

- Float
  - This data type is used to store numbers with decimals.
  - Value without decimal can also be stored in it.
  - Range between 3.4e - 38 to 3.4e +38
  - Occupies 4 byte (32 bits)
  - Format code is %f
- Example,

```
float a,b,c;
a=10.50;
b=20;
c=a+b;
```

C Data Type Introduction:

- Character
  - This data type is used to store alphabets, number and special symbols in it.
  - Range between -128 to +127
    - (Range in ASCII)
  - Occupies 1 byte (8 bits)
  - Format code is %c
- Example,

```
float a,b,c;
a=10.50;
b=20;
c=a+b;
```

C Data Type Introduction:

- Double
  - This data type is used to store numbers with decimals when float is insufficient.
  - Range between 1.7e - 308 to 1.7e +308
  - Occupies 8 byte (32 bits)
  - Format code is %lf
- Example,

```
double a,b,c;
a=9990990.90;
b=20.50;
c=a+b;
```

C Data Type Introduction:

- Void
  - Actually there is no variable defined of this type but it is used do define function.
  - Void is data type which specifies that the function does not return any value to the calling program.
- Example,

```
void main( )
void printnum( )
```

Data Type Modifier

- There are some modifiers which actually change meaning of the primary data type that are available in C as given.
  - Signed
  - Unsigned
  - Long
  - Short

Data Type Modifier

- Signed
  - By default all the data types are treated as signed. So it is not necessary to define any variable as signed.
- Example :

```
signed int a=20;
int b=30;
int c=a+b;
clrscr();
printf("%d",c);
```

Data Type Modifier

- Unsigned
  - If we wants to store only positive values in variable, then it is very easy by using **unsigned**.
  - If we define any variable as unsigned the positive range will be increased of that variable by negative value.
- Example :
  - **integer** value will store **-32768 to +32767** in signed
  - But if we will convert it in to **unsigned** it will store directly **0 to +65535 positive value.**

Data Type Modifier

- Long:
  - If we want to store the bigger value then the data types original range, we can use the long.
- Short:
  - If we want to store small value then the original data types range, we can use the short modifier we can store small value then the original range.

- Generally the data type specifies two things
  1. Range of values that it can store.
  2. Memory requirement to store a data.

Data Type	Key word	Memory Bytes	Range	Format SpeciFier
Character	char	1	-128 to +127	%c
Integer (signed)	int	2	-32768 to +32767	%d
Float (signed)	Float	4	-3.4e38 to +3.4e38	%f
Double	double	8	-1.7e308 to +1.7e308	%lf

Data Type	Key word	Memory Bytes	Range	Format SpeciFier
Long Integer	long	4	21,47,74,83,648 to 21,47,439,647	%ld
Long Double	long double	10	-1.7e932 to +1.7e932	%Lf
Character (unsigned)	Float	4	-3.4e38 to +3.4e38	%f
Double	unsigned char	1	0 to 255	%d

User Defined Type

- In C programming language, users can define their own data types for required variables.
- The user defined type identifier can be used to declare variables.

(1) typedef

- Syntax :

  typedef <data type> <Identifier Name>

- Example :

```
typedef int result;
result s1,s2,s3,s4;
```

User Defined Type

(2) enumerated (enum)

- The enumerated data type is an user defined type.
- This type is used to select one of the serial value enclosed within the braces.
- Syntax :

```
enum <identifier name>{val1, val2... valN};
```

User Defined Type

- Example :

```
enum day{Sun=1,Mon,Tue,Wed,Thu,Fri,Sat};
void main()
{enum day d1;
  d1=Wed;
  printf("\nDay : %d",d1);
}
```

Type Casting :

- Type casting is a process which converts a value from one data type into another data type.
- During the programming **we can convert integer value to float value** and **float value to integer value** as per requirement.
- Example :

```
int a=20;
printf("%f",(float)a/3); //  It will convert data from integer to float.
```

C Pre-Processor

- What is C preprocessor?
  - The C preprocessor is the first part of the compilation process.
- C pre-processor are predefined directives.
- C preprocessor lines beginning with '#'.

C Pre-Processor

No	Directive	Use
1	#include	Specifies the files to be included
2	#define	Defines a macro substitution
3	#undef	Undefines a macro
4	#ifdef	Test for a macro definition
5	#endif	Specifies the end of #if
6	#ifndef	Tests whether a macro is not defined
7	#if	Test a compile-time condition
8	#else	Specifies alternatives when #if test fails

Escape Sequence Character

- Escape sequence characters are the combinations of a backslash (\) followed by a letter or by a combination. These are also known as backslash character constant.

\n	A new line character	\a	To generate sound (bell)
\t	Horizontal tab	\r	Carriage return
\b	Backspace character	\v	Vertical tab
\”	Print double quote	\’	Print single quote
\\	Print single backslash	‘\0’	NULL

- Introduction Of Logic Development Tools

Introduction of Logic :

- Logic is must to perform any task.
- Using various types of logic we can perform many programs.
- There are three types of Logic structures are possible.
  - Sequence Logic
  - Decision Logic
  - Looping Logic

Introduction of Logic :

- Sequence Logic.
  - In this logic all the instructions are written in order to performed in a sequence.
- Decision Logic
  - When more then one option is available then we can use decision logic with.
    - If… Then
    - If… Then…else
    - If… elseif… else… endif

Introduction of Logic :

- Looping Logic :
  - When one or more instructions may be executed more then one time then we can use looping logic.

Instructions for Logic Development

- To develop a program easily, we must have sufficient knowledge of Pre Programming Techniques.
- Pre Programming Techniques will help you to design a proper programming channel.
- It will display scope of errors.
- It will display the chance of development.
- If we are going to write a computer program using any programming language, we need to write a rough program first.

Instructions for Logic Development

- There are two basic tools do develop basic or rough program.
  - Algorithm
  - Flowchart
- If we want to solve any error from any program the we must know the proper steps involved in solving problem.
- If the sequence of programming command is not proper then you will get wrong output.
- To solve these problems we can use Algorithm and Flowchart.

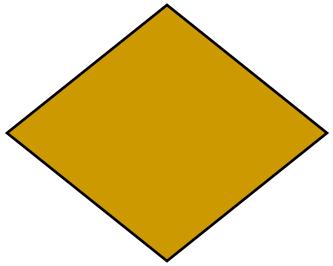
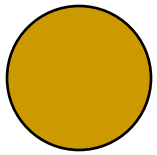
Instructions for Logic Development

- What is Algorithm?
  - Write program solution step by step in textual form is known as algorithm.
- What is Flowchart?
  - A flowchart is a graphical representation of an algorithm.

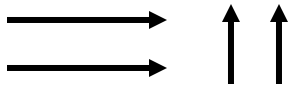
Flowchart Symbols :

	**Terminal Symbol (Start / Stop)** This oval shape represents for Start and Stop.
	**Process Steps Symbol** The Rectangle represents the processing operation.
	**Input or Output Process Symbol** This shape represents input and output task.

Flowchart Symbols :

	Decision Making and Branching Symbol The diamond shape represent decision or branching statements.
	Connector Symbol The circle represents a logical flow from one page of flowchart to another page.

Flowchart Symbols :

	**Flow Direction Symbol** The arrows shows flow direction in which one has to proceed to solve the problem.
	**Predefined Process** This oval shape represents for Start and Stop.

Sequence Logic

Algorithm :

1. Write an algorithm and draw a flowchart to read and print value.

   Step-1 Input the number

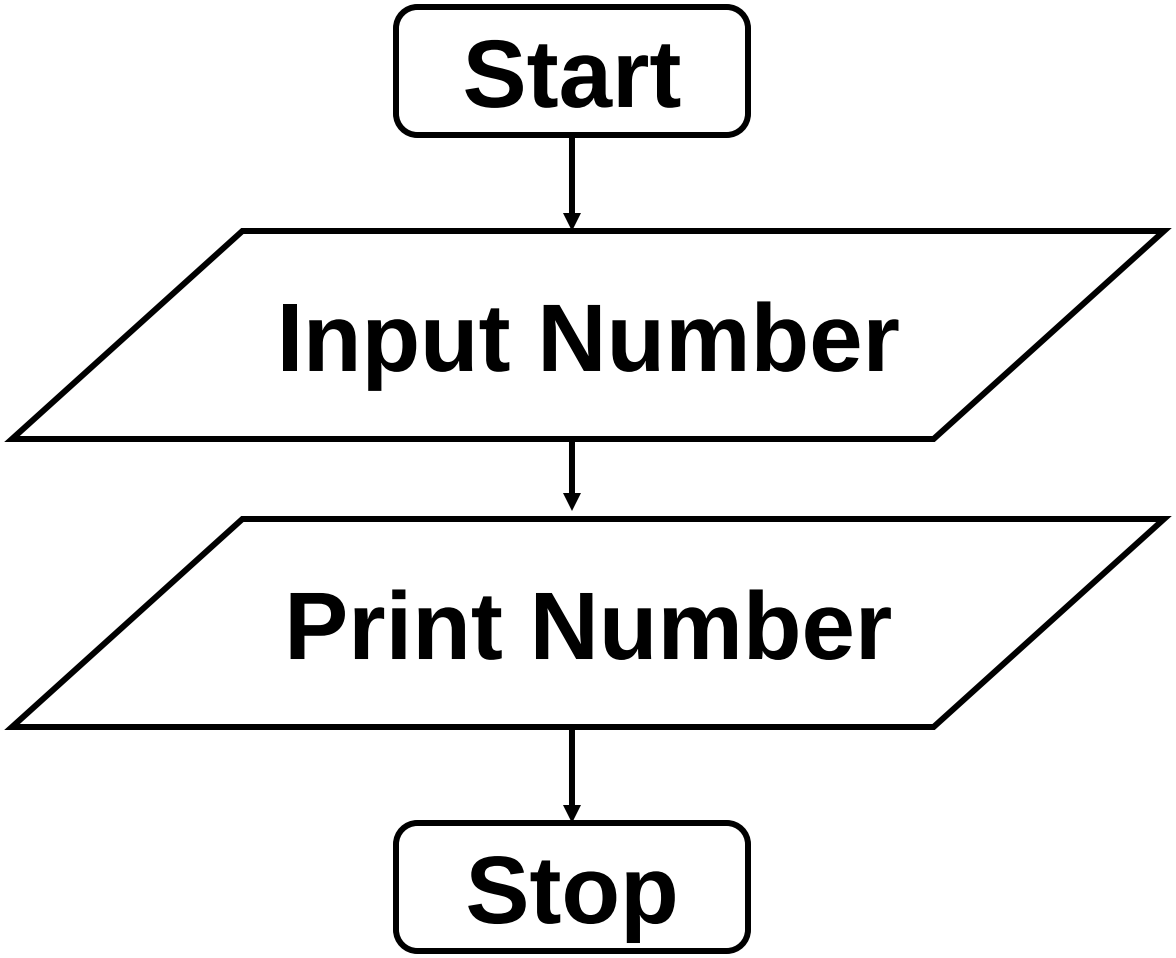
   Step-2 Print the number

   Step-3 Stop

Sequence Logic

Flowchart :

1. Draw a flowchart to read and print value.



Sequence Logic

Algorithm :

2. Write an algorithm to make sum of three value.

Step-1 Input value1, value2, value3

Step-2 Calculate sum of three values

Sum=value1+value2+value3

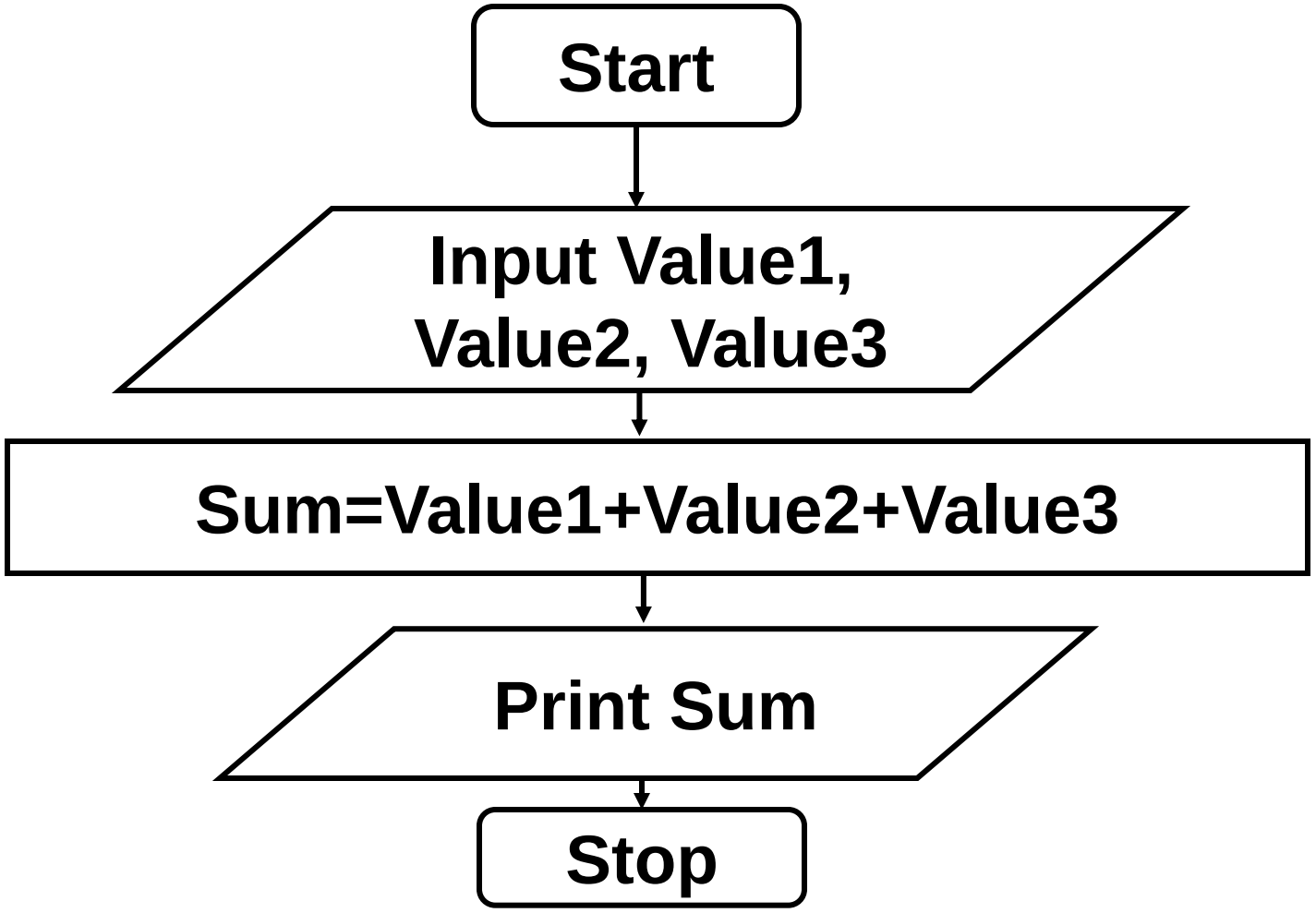
Step-3 Print sum of value

Step-4 Stop

Sequence Logic

Flowchart :

2. Draw a flowchart to make sum of three value.



Decision Logic

Algorithm :

- Write an algorithm to find the maximum number from given two number.

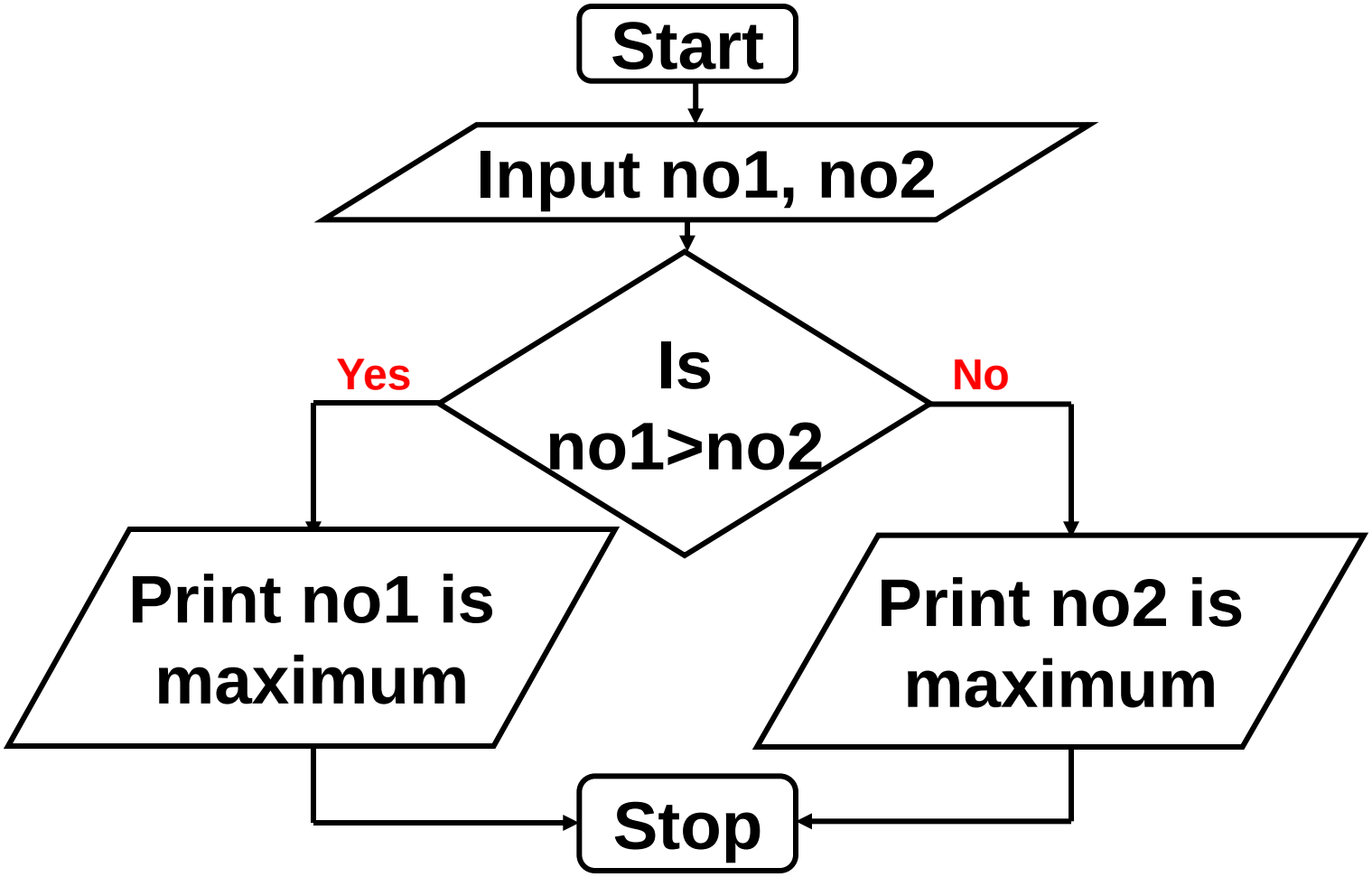
Step-1 Input value of no1 and no2

Step-2 If no1>no2, if yes then print no1 as maximum else no2 as maximum

Step-3 Stop

Decision Logic

Flowchart : Draw a flowchart to find the maximum number from given two number.



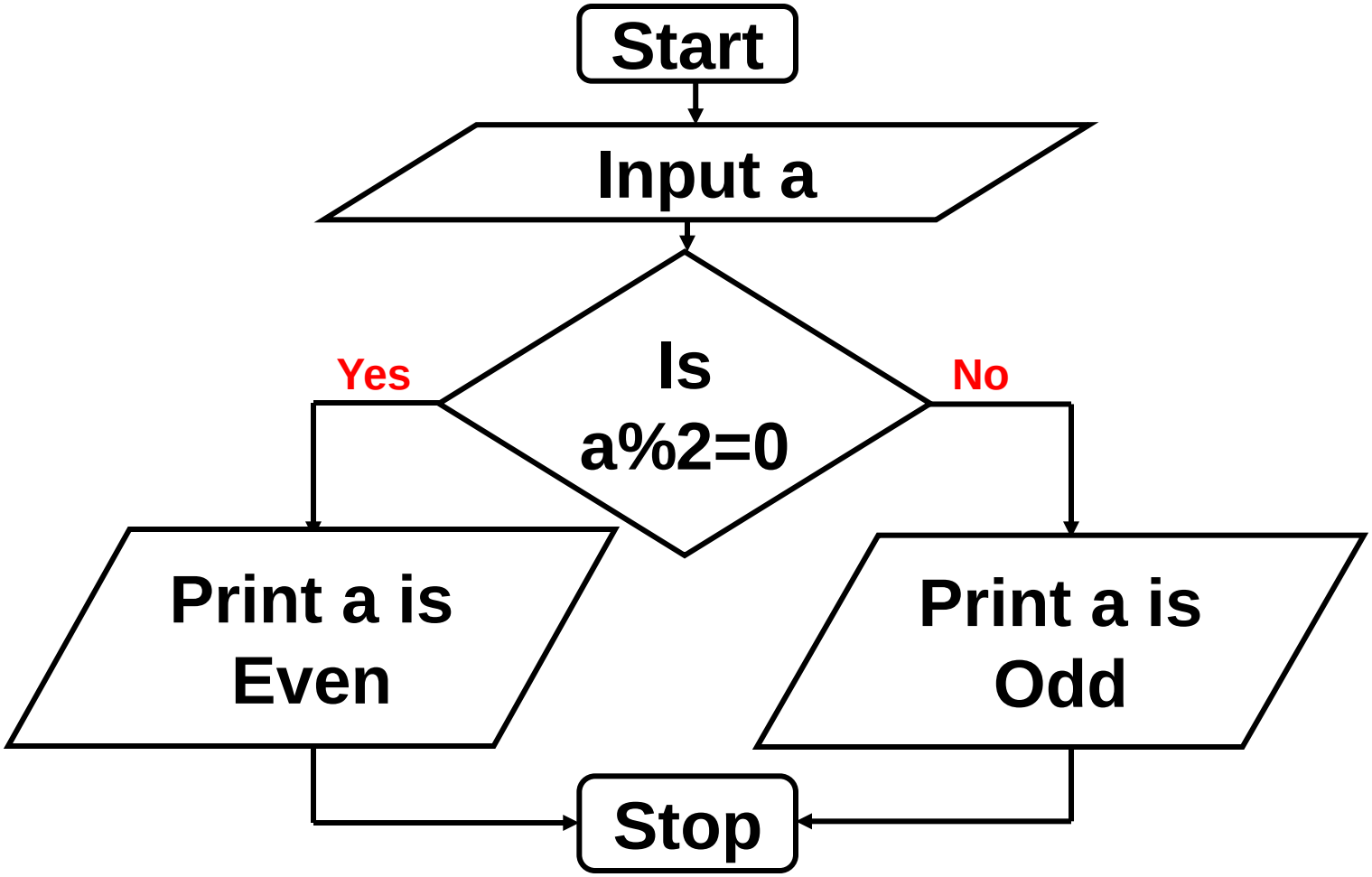
Decision Logic

Algorithm :

- Write an algorithm and draw a flowchart to fine that given value is odd(એકી) or even(બેકી)
  - Step-1 Input value
  - Step-2 Check if value%2=0, if yes then value is EVEN else value is ODD
  - Step-3 Stop

Decision Logic

Flowchart : Draw a flowchart to find that given value is odd(એકી) or even(બેકી).



Decision Logic

- **Algorithm :** Write an algorithm to find grade for given percent.

Step-1 Input **per**

Step-2 If **per**>=70 if yes then print DISTINCTION else goto step three

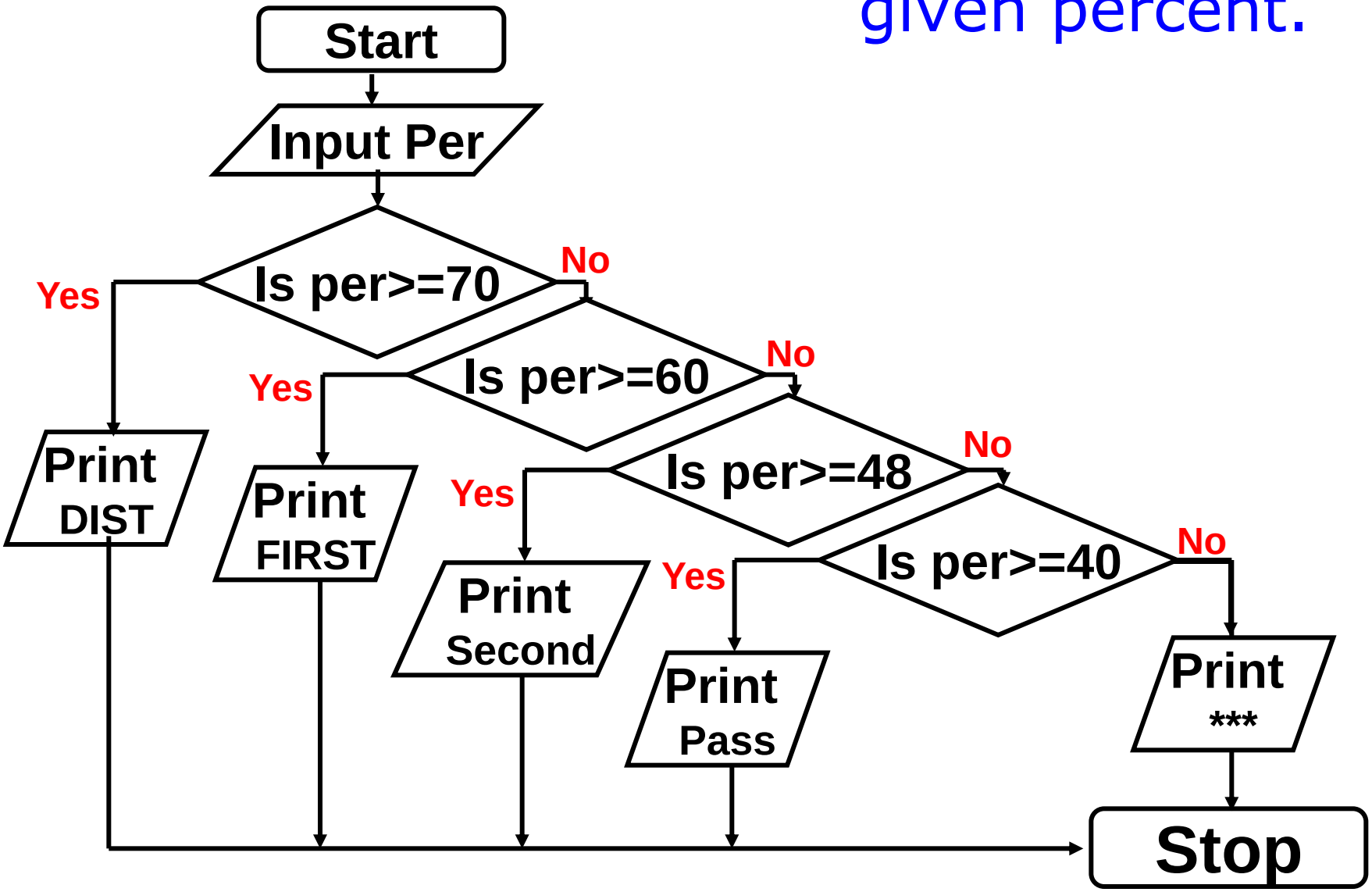
Step-3 If **per**>=60 if yes then print FIRST else goto step four

Step-4 If **per**>=48 if yes then print SECOND else goto step five

Step-5 If **per**>=40 if yes then print PASS else print ***

Decision Logic

Flowchart : Draw a flowchart to find grade for given percent.



Looping Logic

- **Algorithm :** Write an algorithm to print 1 to 10 serial number

Step-1 Input i=1

Step-2 Check is i<=10, if yes then goto Step-3 else goto Step-5

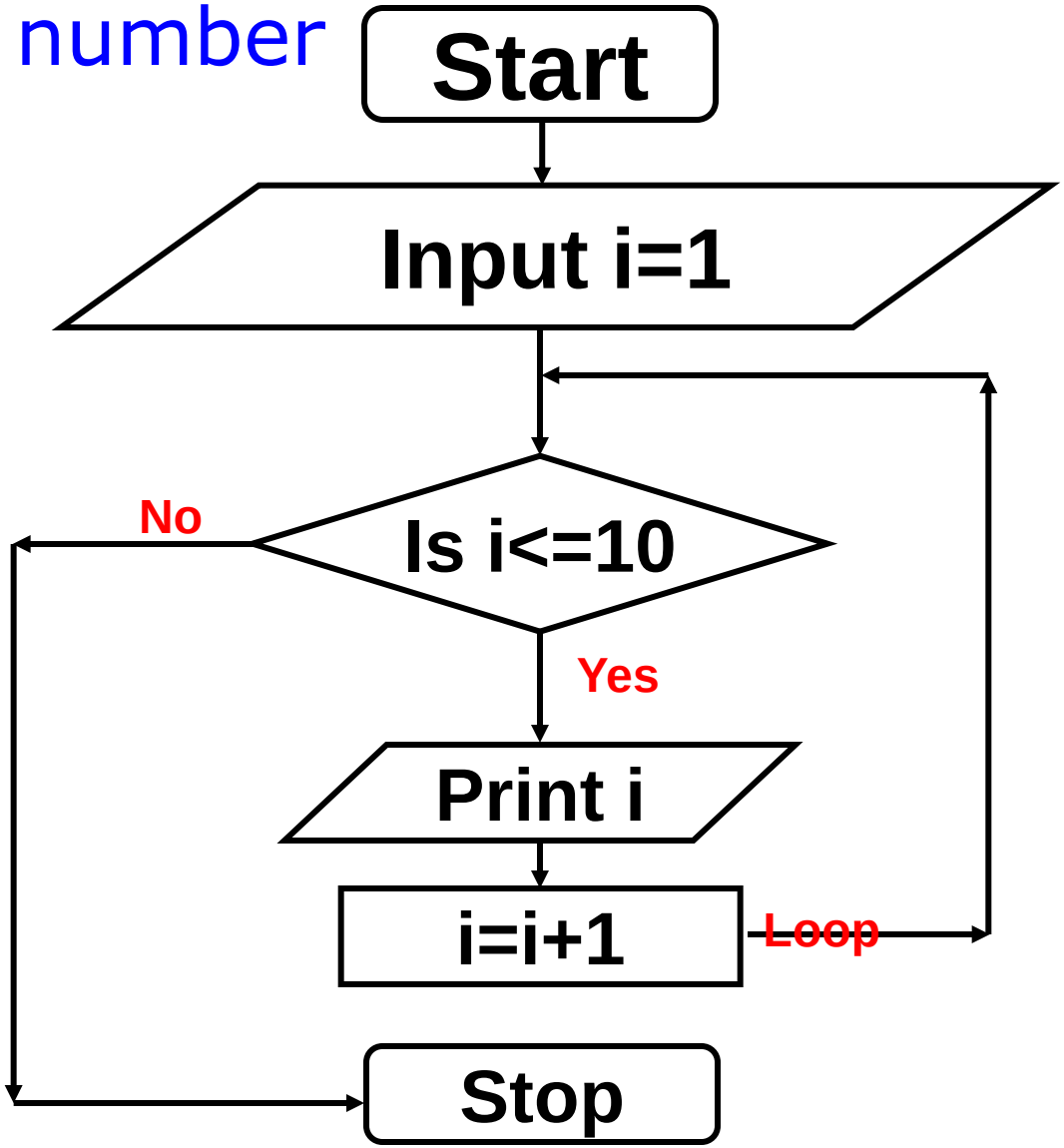
Step-3 Print I

Step-4 i=i+1, goto Step-2

Setp-5 Stop

Decision Logic

Flowchart : Draw a flowchart to print 1 to 10 serial number



What is dry-run?

- In computer programming, a dry run is a mental run of a computer program.
- When the computer programmer examine the source code a step at a time and determines what it will do when run.

Use of dry run in programming :

- To know the status of variable at every stop.
- To find out error in any inner part or block of the program.
- To check any syntax or logical error from the program.
- To know the order of execution process of program dry run is very useful.

Other Logic Development Techniques

- There are many techniques used for developing program logic to solve any problem definition.
- We already discussed about **algorithm** and **flowchart**.
- Modular programming concept using user defined function.
- To check program execution process using dry run.
- To use watch and output window features of C editor to debug program step by step for analyzing internal process of the program.