UNIT 1 : INTRODUCTION

1.1 Concepts of Programming Language

- 1.1.1 Introduction of Source Code, Object Code and executable code
- 1.1.2 Algorithm and Flowchart
- 1.1.3 Concepts of Structured Programming Language

1.2 Concepts of Editor, Interpreter and Compiler

- 1.2.1 Introduction of C program body structure
- 1.2.2 Character Set, concepts of variables and constants
- 1.2.3 Identifiers, literals, Key words
- 1.2.4 Data types (signed and unsigned) (Numeric : int, short int, long, float, double) , (Character type: char, string) and void.
- 1.2.5 Concepts of source code, object code and executable code

1.1 Concepts of Programming Language

1.1.1 Introduction of Source Code, Object Code and executable code

1.1.2 Algorithm and Flowchart

1.1.3 Concepts of Structured Programming Language

Programming Language

- A programming language is a formal language comprising a set of instructions that produce various kinds of output.
- Programming languages are used in computer programming to implement algorithms.
- Most programming languages consist of instructions for computers.

1.1.1 Introduction of Source Code, Object Code and Executable code

1.1.2 Algorithm and Flowchart

Algorithm

- The **step by step procedure** for **solving a problem** is known as algorithm.
- A sequence of **precise** and **unambiguous** instructions for solving a problem is known as algorithm.

Example:

- Write an Algorithm to make tea.
- Step 1 : START / BEGIN
- Step 2 : Turn on Gas
- Step 3 : Take Bowl/Jar
- Step 4 : Add Milk, Water, Sugar, Tea, MASALA
- Step 5 : Boil it for sometime
- Step 6 : Turn off Gas
- Step 7 : STOP / END

Example:

Write an algorithm to make tea.

```
Step 1: START / BEGIN
Step 2: Take a bowl
Step 3: Add water and Milk in bowl
Step 4: Light up Gas
Step 5: Put Bowl on Gas
Step 6: Add Sugar and Tea
Step 7: Boil for Some Time
Step 8: Turn off Gas
Step 9: STOP / END
```

Example:

Write an algorithm to eat Pizza in Lockdown.

Step 1 : START / BEGIN

Step 2 : Ask Parents for permission

Step 3 : They said no.

Step 4 :

Step 4 : STOP / END

Homework:

1. Write an Algorithm to go for movie with friend.
2. Write an Algorithm to go college from home.
3. Write an Algorithm to picnic with friends.
4. Write an Algorithm to print your name.
5. Write an Algorithm to print addition of two numbers.

- Write an Algorithm to print your name.
- Step 1 : START
- Step 2 : Take an element X
- Step 3 : Store your name in X
- Step 4 : Print / Display X
- Step 5 : STOP

- Write an Algorithm to print your name.
- Step 1 : START
- Step 2 : DISPLAY / PRINT / WRITE “CHIRAG”
- Step 3 : STOP

- Write an Algorithm to print your name, Address, Phone Num.
- Step 1 : START
- Step 2 : DISPLAY / PRINT / WRITE "CHIRAG" , "SURAT", "9913…."
- Step 3 : DISPLAY "SURAT"
- Step 4 : DISPLAY "9913776671"
- Step 5 : STOP

- Write an Alogrithm to print Your 12th STD Marksheet.

- Write an Algorithm to print addition of two numbers.
- Step 1 : START
- Step 2 : Declare a=10 and Declare b=10
- Step 3 : Calculate c = a+b
- Step 4 : Display c
- Step 5 : STOP

- Write an Algorithm to print cube of number.
- Step 1 : START
- Step 2 : Declare a=10
- Step 3 : Calculate c = a*a*a
- Step 4 : Display c
- Step 5 : STOP

- Write an Alogrithm to print Your 12th STD Marksheet.
- Write an Algorithm to print Addition(+), Subtraction(-), Multiplication(*) and Division(/) of two numbers.
- Write an Algorithm to print Cube of given number.

Sample Calculation

Declare a=10	c=a*a*a	Display c
a=10	c=10*10*10=100	1000
a=5	c=5*5=25	25
a=12		

- Step 1 : START
- Step 2 : Declare name=“VTCBB” and seatnum=“20BCA151”
- Step 3 : Declare cs=90,maths=99,ic=90,cppm=36,dma=90
- Step 4 : Display name, Seatnum
- Step 5 : Display cs,maths,ic,cppm,dma
- Step 6 : Calculate t=cs+maths+ic+cppm+dma
- Step 7 : Calculate per=t/5 or t*100/500
- Step 8 : Display t and per.
- Step 9 : Stop

Declare name="VTCBB" and seatnum="20BCA151"	Declare cs=90,maths=99,ic=90,cppm=36,dma=90	Display name, Seat num	Display cs,maths,ic,cppm,dma	Calculate t=cs+maths+ic+cppm+dma	Calculate per=t/5	Display t and per
Name="VTCBB" Seat num = 20BCA151	cs=90 Maths=99 Ic=90 Cppm=36 Dma=90	VTCBB 20BCA151	90 99 90 36 90	t=90+99+90+36+90 =405	per=405/5 =81	405 81

Write an Algorithm to print Addition(+), Subtraction(-), Multiplication(*) and Division(/) of two numbers.

```
Step 1 : START
Step 2 : Declare a=20,b=10,c=0
Step 3 : Calculate c=a+b
Step 4 : Calculate c=a-b
Step 5 : Calculate c=a*b
Step 6 : Calculate c=a/b
Step 7 : Display c
Step 8 : STOP
```

Declare a=20,b=10, c=0	**c=a+b**	**c=a-b**	**c=a*b**	**c=a/b**	**Display c**
A=20 B=10 C=0	C=20+10 =30	C=20-10 =10	C=20*10 =200	C=20/10 =2	2

Write an Algorithm to print Addition(+), Subtraction(-), Multiplication(*) and Division(/) of two numbers.

```
Step 1 : START
Step 2 : Declare a=20,b=10, c=0
Step 3 : Calculate c=a+b
Step 4 : Display c
Step 5 : Calculate c=a-b
Step 6 : Display c
Step 7 : Calculate c=a*b
Step 8 : Display c
Step 9 : Calculate c=a/b
Step 10 : Display c
Step 11 : STOP
```

Declare a=20,b= 10,c=0	c=a+b	Display c	c=a-b	Display c	c=a*b	Display c	c=a/b	Display c
A=20 b-=10 C=0	C=20+10 =30	30	C=20-10 =10	10	C=20*10 =200	200	C=20/10 =2	2

Write an Algorithm to print Addition(+), Subtraction(-), Multiplication(*) and Division(/) of two numbers.

```
Step 1 : START
Step 2 : Declare a=20,b=10, c=0,d=0,e=0,f=0
Step 3 : Calculate c=a+b
Step 4 : Calculate d=a-b
Step 5 : Calculate e=a*b
Step 6 : Calculate f=a/b
Step 7 : Display c,d,e,f
Step 8 : STOP
```

Declare a=20,b=10, c=0,d=0,e= 0,f=0	c=a+b	d=a-b	e=a*b	f=a/b	Display c,d,e,f
A=20 B=10 C=0 D=0 E=0 F=0	c=20+10 =30	D=20-10 =10	E=20*10 =200	F=20/10 =2	30 10 200 2

- Write an Algorithm and Sample Calculation to convert dollar to rs.(1 Dollar = 70 rs)
- Write an Algorithm and Sample Calculation to convert Kilogram to Grams.
- Write an Algorithm and Sample Calculation to convert GB to MB.
- Write an Algorithm and Sample Calculation to Swap (interchange) two numbers

- Write an Algorithm and Sample Calculation to convert rs to dollar.
- Write an Algorithm and Sample Calculation to convert Gram to Kilograms.
- Write an Algorithm and Sample Calculation to convert MB to GB.
- Write an Algorithm and Sample Calculation to Swap (interchange) two numbers.(without using third variable)

- START
- DECLARE D = 2
- R=D*70
- DISPLAY R

- DECLARE K = 2
- G=K*1000
- DISPLAY G

- DECLARE G = 2
- M=G*1024
- DISPLAY M

A=2	R=A*70	DISPLAY R	
A=2	R=2*70 =140	140	
A=5	R=5*70 =350	350	

```
START
DECLARE a=10, b=20, c=0
c=a
a=b
b=c
Display a
Display b
```

Declare a =10 b=20 c=0	a=b	b=a	Display a Display b

A=10
B=20
	A=20	B=20	20
			20

a=10 b=20 temp=0	temp=a	a=b	b=temp	Display a Display b

a=10
b=20
c=0
	c=10	a=20	b=10	A=20
				B=10

A=10 **B=20** **Temp=0**	**Temp=**			

- a=10 , b=20
- ..
- …
- ..
- Display a   // 20
- Display b // 10

Write an Algorithm and Sample Calculation to convert rs to dollar.

- DECLARE D=2
- R=D*70
- DISPLAY R

- START
- DECLARE R=140
- D = R/70
- DISPLAY D

- R=2
- D=R*0.014
- DISPLAY D

- START
- DECLARE R=140
- D = R/70
- DISPLAY D

- START
- DECLARE G=14000
- K = G/1000
- DISPLAY K

- START
- DECLARE MB=2048
- GB = MB/1024
- DISPLAY GB

- DECLARE A=10, B=20
- A=A+B
- B=A-B
- A=A-B
- DISPLAY A,B

A=10
B=20
	A=A*B	B=A/B	A=A/B	DISPLAY
				A
				B

A=10
B=20
	A=10+20
	30
		B=30-20
		=10
			A=30-10
			=20
				20
				10

A=30
B=50
	A=30+50
	=80
		B=80-50
		=30
			A=80-30
			=50
				50
				30

A=10
B=20
	A=10*20
	=200
		B=200/20
		=10
			A=200/10
			=20
				20
				10

Advantages of Algorithms:

- It is a **step-wise representation** of a solution to a given problem, which makes it **easy to understand**.
- An algorithm uses a **definite procedure**.
- It is **not dependent** on any programming language, so it is easy to understand for anyone even **without programming knowledge**.
- 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** hence, it is easier for programmer to convert it into an actual program.

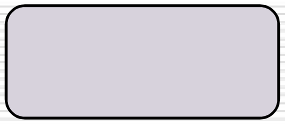
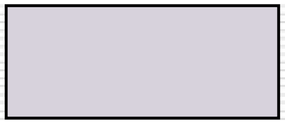
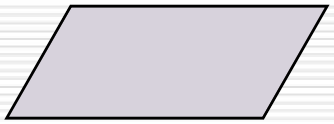
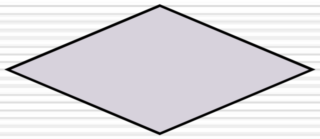
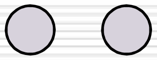
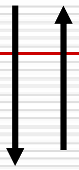
Disdvantages of Algorithms:

- Algorithms is **Time consuming**.
- Difficult to **show Branching and Looping** in Algorithms.
- **Big tasks** are **difficult to put** in Algorithms.

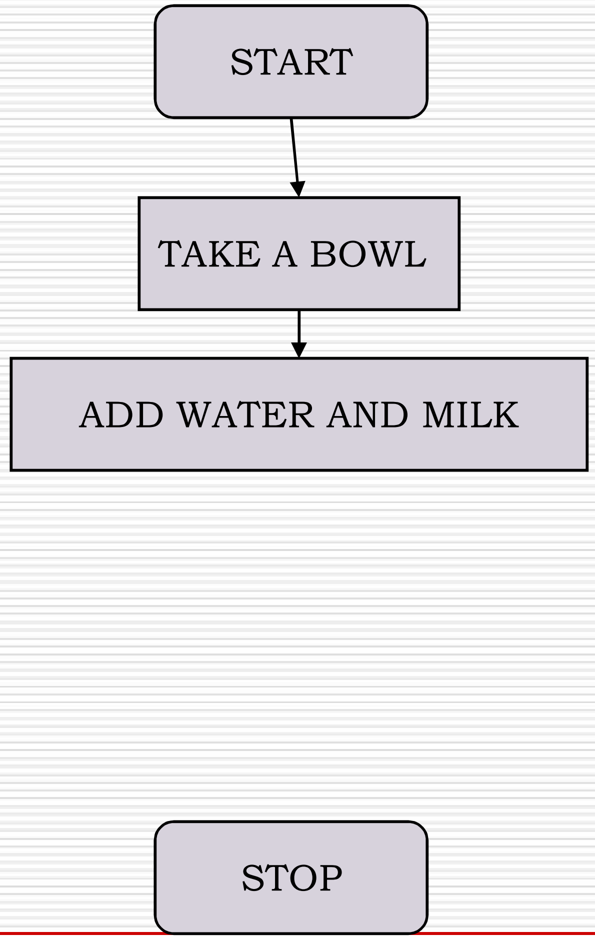
Flowchart

- A flowchart is a **diagrammatic representation** that illustrates the **sequence of operations** to be performed to **arrive at the solution**.
- A **pictorial representation**, which uses **predefined symbols**, to describe either the **logic** of a computer program (program flowchart), or the **data flow** and processing step of a system.
- A graphical representation of an algorithm is known as flowchart.

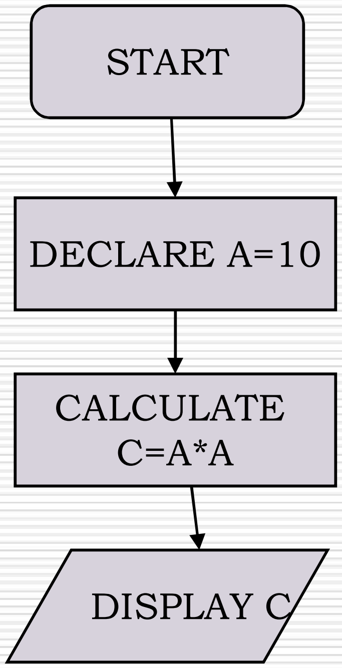
Symbols used in Flowchart

	**Start or End of the program**
	**Process / Computational Steps**
	**Input/Output Instructions**
	**Decision making & Branching**
	**Connectors**
	**Flow Line**

Example: Draw a flowchart to make tea



DRAW FLOWCHART TO PRINT ADDITION OF TWO NUMBERS



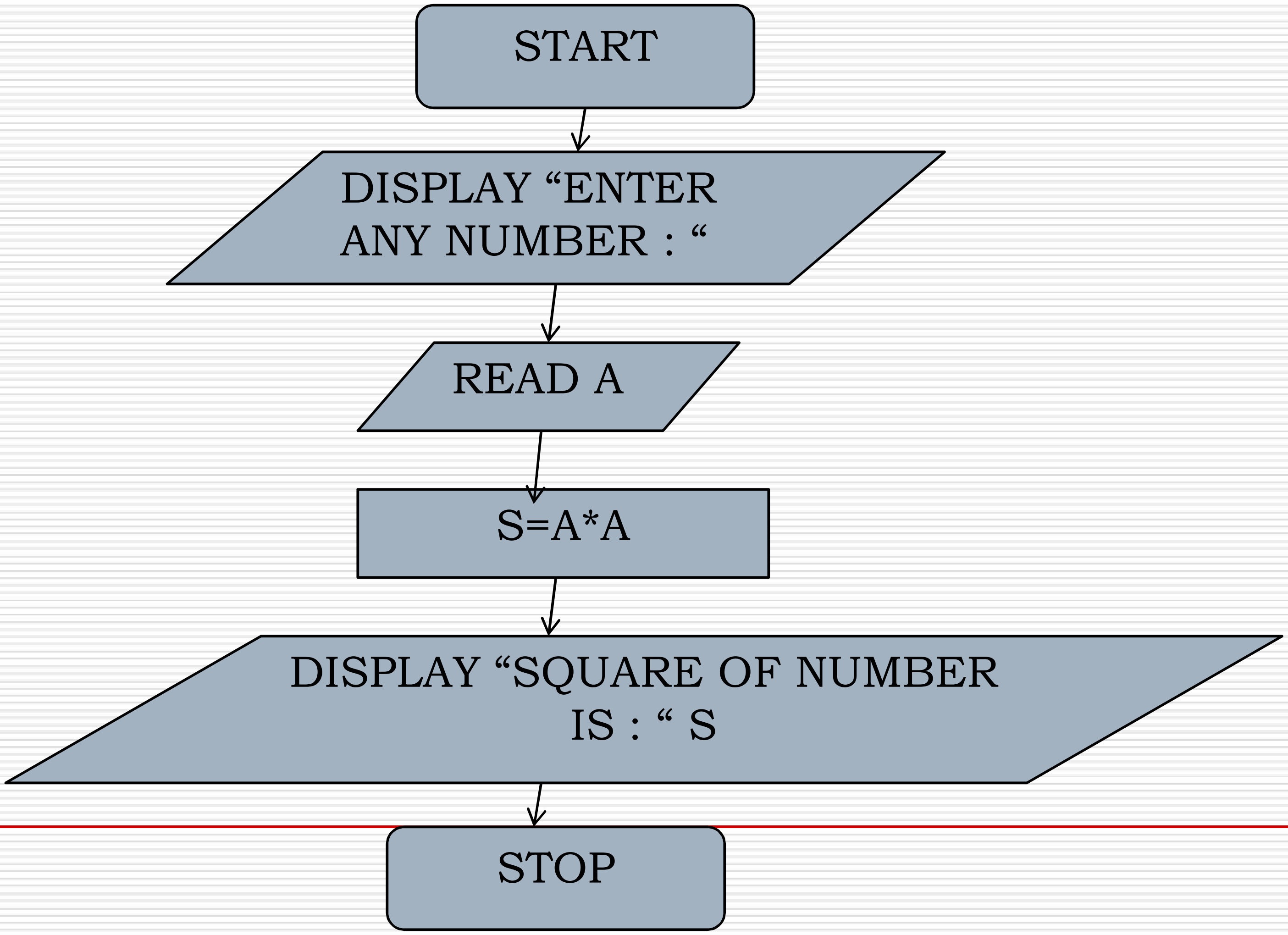
DRAW ALGORITHM TO PRINT SQUARE OF GIVEN NUMBER.

- STEP 1 : START
- STEP 2 : DISPLAY “ENTER ANY NUMBER:”
- STEP 3 : READ A / INPUT A
- STEP 4 : CALCULATE S=A*A
- STEP 5 : DISPLAY “SQUARE OF NUMBER IS : “ S
- STEP 6 : STOP

SAMPLE CALCULATION

DISPLAY "ENTER ANY NUMBER"	READ A	S=A*A	DISPLAY "SQUARE OF NUMBER IS : " S
ENTER ANY NUMBER	A=5	S=5*5 =25	SQUARE OF NUMBER IS : 25
ENTER ANY NUMBER	A=25	S=25*25 =625	SQUARE OF NUMBER IS : 625

DRAW FLOWCHART TO PRINT SQUARE OF GIVEN NUMBER


START
DISPLAY “ENTER ANY NUMBER : “
READ A
S=A*A
DISPLAY “SQUARE OF NUMBER IS : “ S
STOP

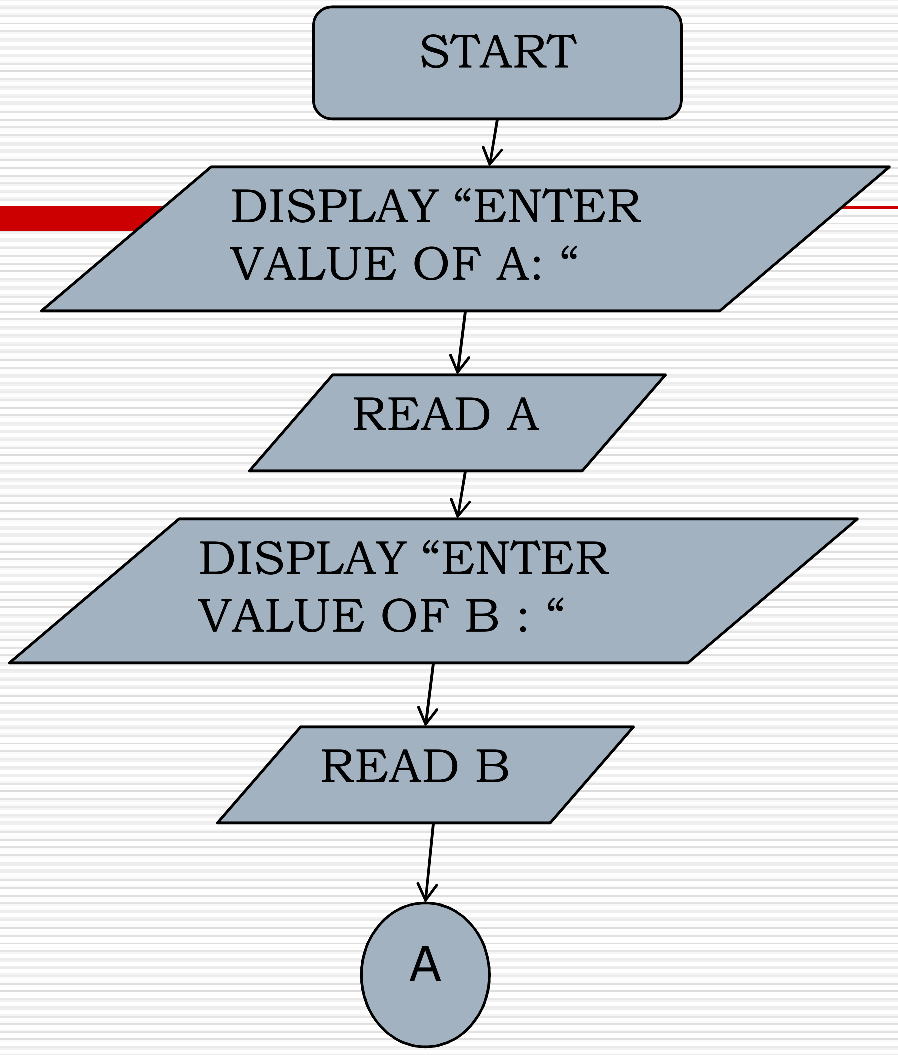
HOMEWORK(WRITE ALGORITHM, DRAW FLOWCHART AND SAMPLE CALCULATION(MIN. 3 VALUES) FOR THE FOLLOWING:

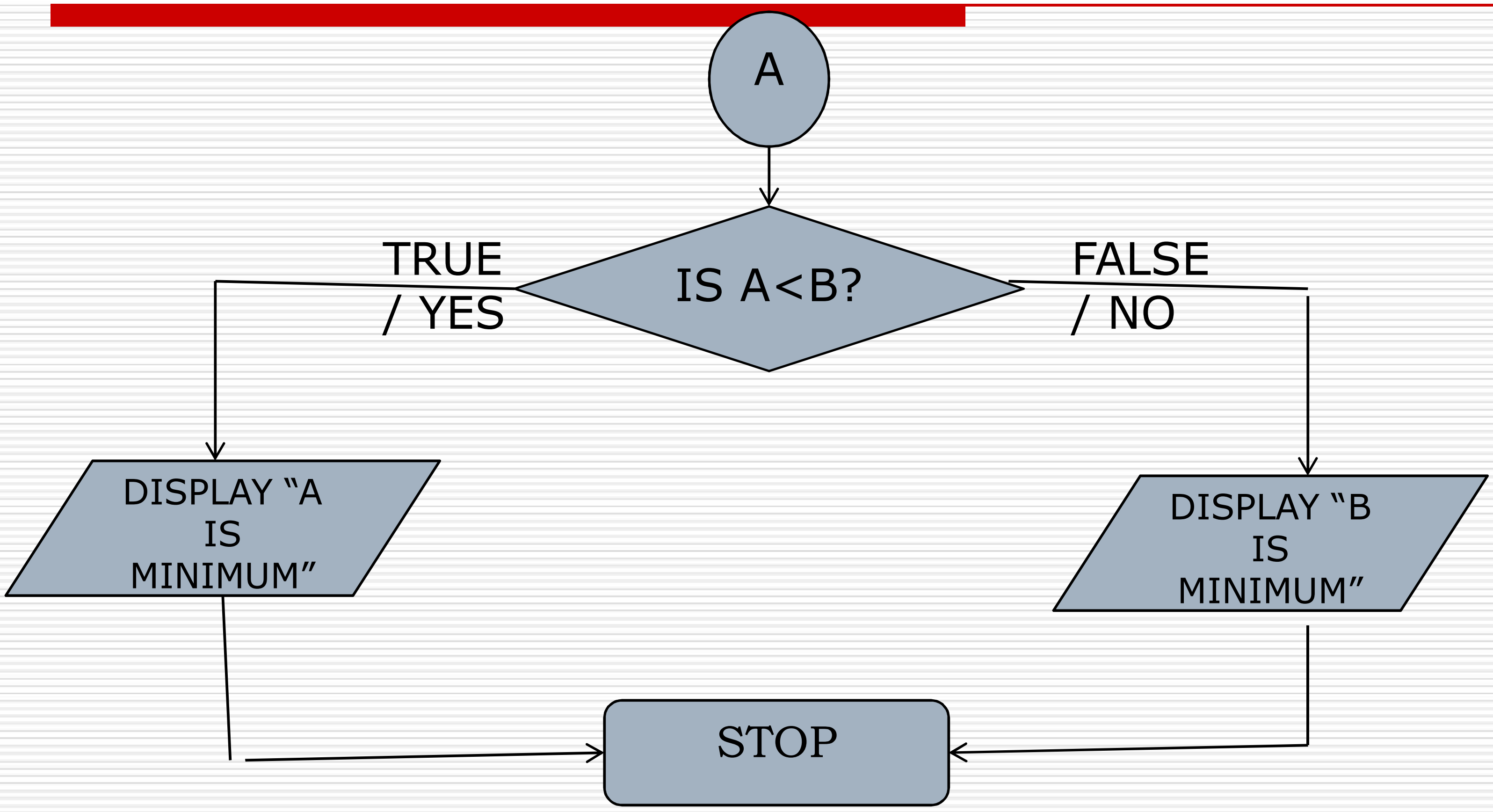
1. TO PRINT YOUR NAME, ADDRESS, PHONE NUMBER
2. TO PRINT ADDITION OF 2 NUMBERS
3. TO PRINT SUBSTRACTION OF 2 NUMBERS
4. TO PRINT MULTIPLICATION OF 2 NUMBERS
5. TO PRINT DIVISION OF 2 NUMBERS
6. TO PRINT CUBE OF GIVEN NUMBER
7. TO CONVERT DOLLAR TO RS
8. TO CONVERT RS TO DOLLAR
9. TO CONVERT MB TO GB
10. TO SWAP TWO NUMBERS

WRITE AN ALGORITHM, FLOWCHART, SAMPLE CALCULATION TO FIND MINIMUM OF TWO NUMBERS

```
STEP 1 : START
STEP 2 : DISPLAY “ENTER VALUE OF A”
STEP 3 : READ A
STEP 4 : DISPLAY “ENTER VALUE OF B”
STEP 5 : READ B
STEP 6 : IF A<B THEN GO TO STEP 7 ELSE GO TO STEP 8
STEP 7 : DISPLAY “A IS MINIMUM”
STEP 8 : DISPLAY “B IS MINIMUM”
STEP 9 : STOP
```

DISPLAY "ENTER VALUE OF A"	READ A	DISPLAY "ENTER VALUE OF B"	READ B	IS A< B ?	DISPLAY "A IS MINIMUM"	DISPLAY "B IS MINIMUM"
ENTER VALUE OF A	A=10	ENTER VALUE OF B	B=20	IS 10<20? TRUE	A IS MINIMUM	---------
ENTER VALUE OF A	A=20	ENTER VALUE OF B	B=15	IS 20<15? FALSE	---------	B IS MINIMUM


START
DISPLAY "ENTER VALUE OF A: "
READ A
DISPLAY "ENTER VALUE OF B : "
READ B
A


A
IS A<B?
TRUE / YES
FALSE / NO
DISPLAY “A IS MINIMUM”
DISPLAY “B IS MINIMUM”
STOP

ASSIGNMENT -1

- What is Algorithm? Write advantages and disadvantages of it.
- What is Flowchart? Write advantages and disadvantages of it.
- What is Flowchart? Discuss its symbols with proper example.

- Write a Program to print Subtraction of 2 numbers.
- Write a Program to print Multiplication of 2 numbers.
- Write a Program to print Division of 2 numbers.
- Write a Program to print Addition, Subtraction, Multiplication and Division of 2 numbers.
- Write a Program to print Square of numbers.

Advantages:

- **Communication**

  Flowchart are better way of communicating the logic of the system

- **Effective analysis**

  With the help of flowchart, a problem can be analyzed in a more effective way.

- **Efficient Coding**

  The flowchart act as a guide or blue print during the system analysis and program development phase.

- **Proper debugging**

  The flowchart helps in debugging process. Debugging means finding and correcting problems or mistakes or errors.

- **Proper documentation**

  A flowchart serves as a good program documentation which is needed for future purpose.

- **Efficient program maintenance**

  Using the flowchart, we can easily manage the program or a system,

Disadvantage:

- **Complex logic**

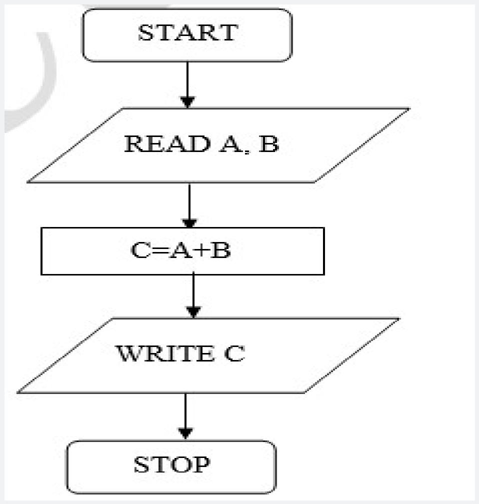
  Sometimes the program logic is complicated in that case flowchart becomes more complex.

- **Alteration and modification**

  If changes are required then the flowchart may require to redraw completely.

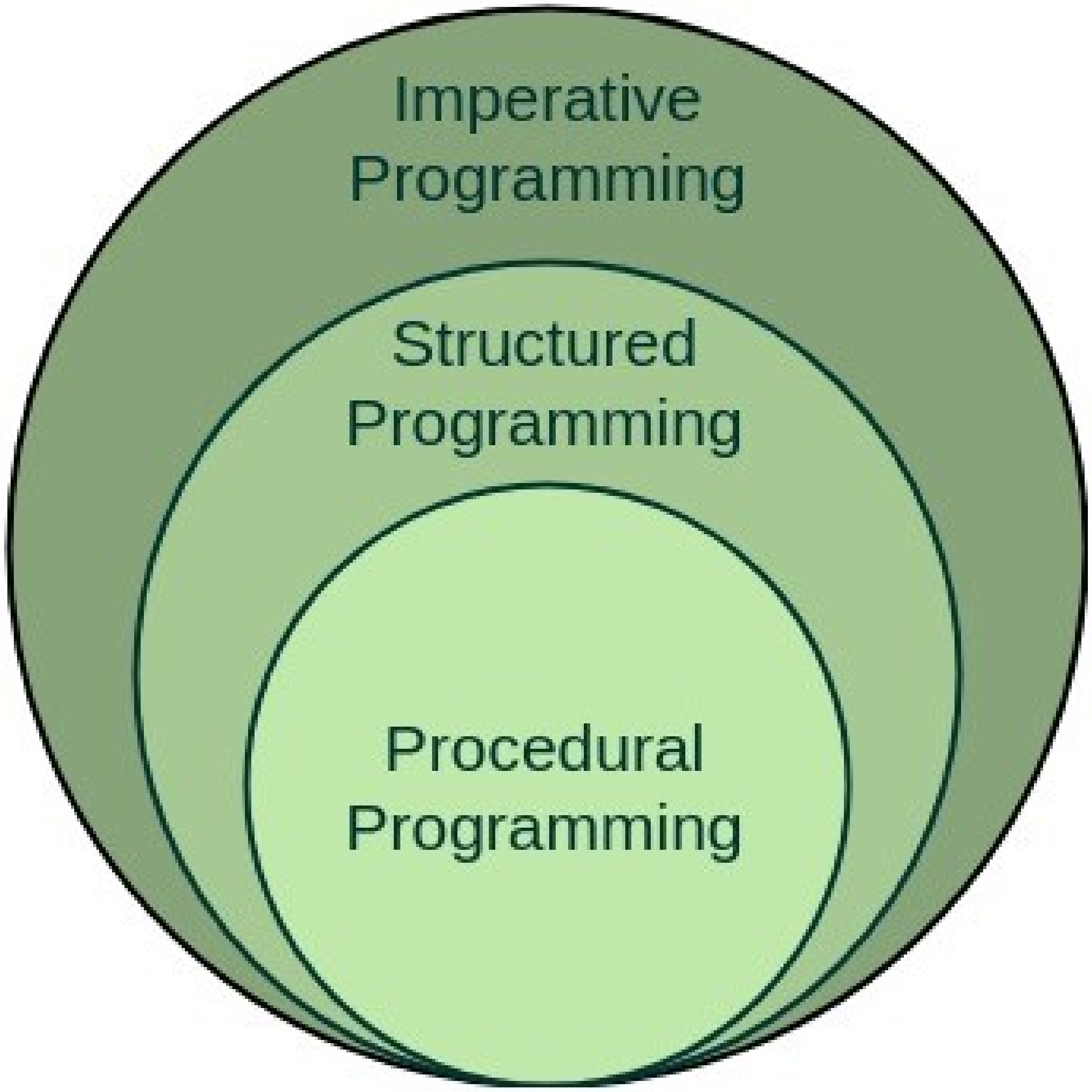
- **Reproduction**

  As the flowchart symbols are not to be typed, reproduction of flowchart may require to draw again.

Algorithm	Flowchart
Algorithm is step by step procedure for solving a problem.	Flowchart is graphical representation of an algorithm.
In algorithm, there are steps and statements.	In flowchart, there are predefined symbols.
Algorithm is difficult to understand.	Flowchart is easy to understand.
Eg: Step 1: START Step 2: READ A and B Step 3: CALCULATE C=A+B Step 4: WRITE C Step 5: STOP	
Difficult to find mistake Algorithm is not user friendly Algorithm is not Time Consuming	Easy to find Mistake Flowchart is user friendly Flowchart is Time Consuming

1.1.3 Concepts of Structured Programming Language

- **Structured Programming Approach**, as the word suggests, can be defined as a programming approach in which the program is made as a **single structure**.
- It means that the code will execute the **instruction by instruction** one after the other.
- It doesn't support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner.
- The languages that support Structured programming approach are:
- C
- C++
- Java
- C#


Imperative Programming
Structured Programming
Procedural Programming

- The structured program consists of **well structured and separated modules**. But the entry and exit in a Structured program is a **single-time** event.
- It means that the program uses **single-entry and single-exit elements**. Therefore a structured program is **well maintained, neat and clean** program.
- This is the reason why the Structured Programming Approach is well accepted in the programming world.

- The structured program mainly consists of three types of elements:
  - Selection Statements
  - Sequence Statements
  - Iteration Statements

Advantages:

- Easier to read and understand
- User Friendly
- Easier to Maintain
- Mainly problem based instead of being machine based
- Development is easier as it requires less effort and time
- Easier to Debug
- Machine-Independent, mostly.

Disadvantages:

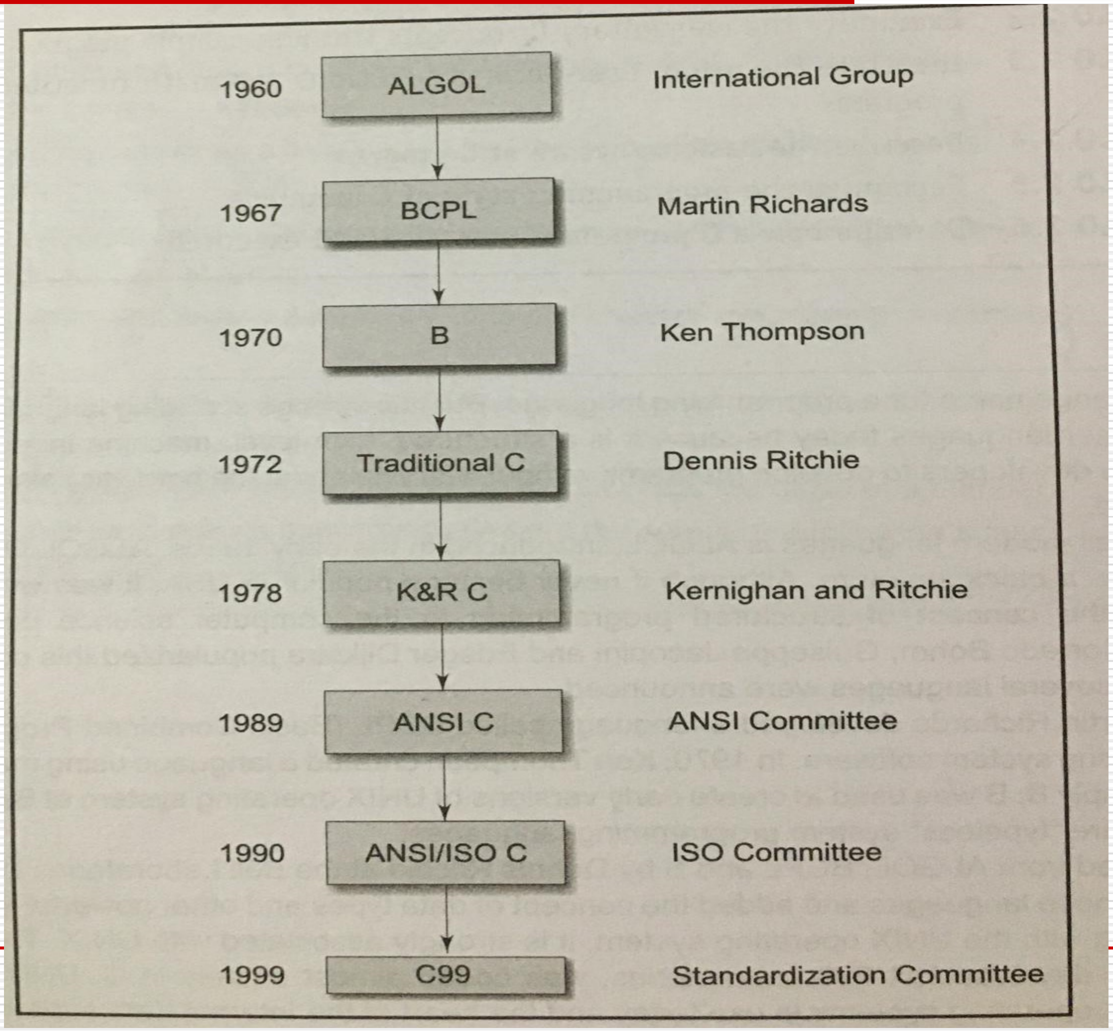
- Since it is Machine-Independent, So it **takes time to convert into machine code**.
- The converted machine code is not the same as for **assembly language**.
- The program depends upon **changeable factors** like data-types. Therefore it needs to be updated with the need on the go.
- Usually the development in this approach takes **longer time** as it is language-dependent. Whereas in the case of assembly language, the development takes lesser time as it is fixed for the machine.

- 1.2 Concepts of Editor, Interpreter and Compiler
  - 1.2.1 Introduction of C program body structure
  - 1.2.2 Character Set, concepts of variables and constants
  - 1.2.3 Identifiers, literals, Key words
  - 1.2.4 Data types (signed and unsigned) (Numeric : int, short int, long, float,
  - double) , (Character type: char, string) and void.
  - 1.2.5 Concepts of source code, object code and executable code

1.2.1 Introduction of C program

- C is a general-purpose high level language that was originally developed by Dennis Ritchie in 1972 at AT & T Bell Lab, for the Unix operating system.

History of 'C'

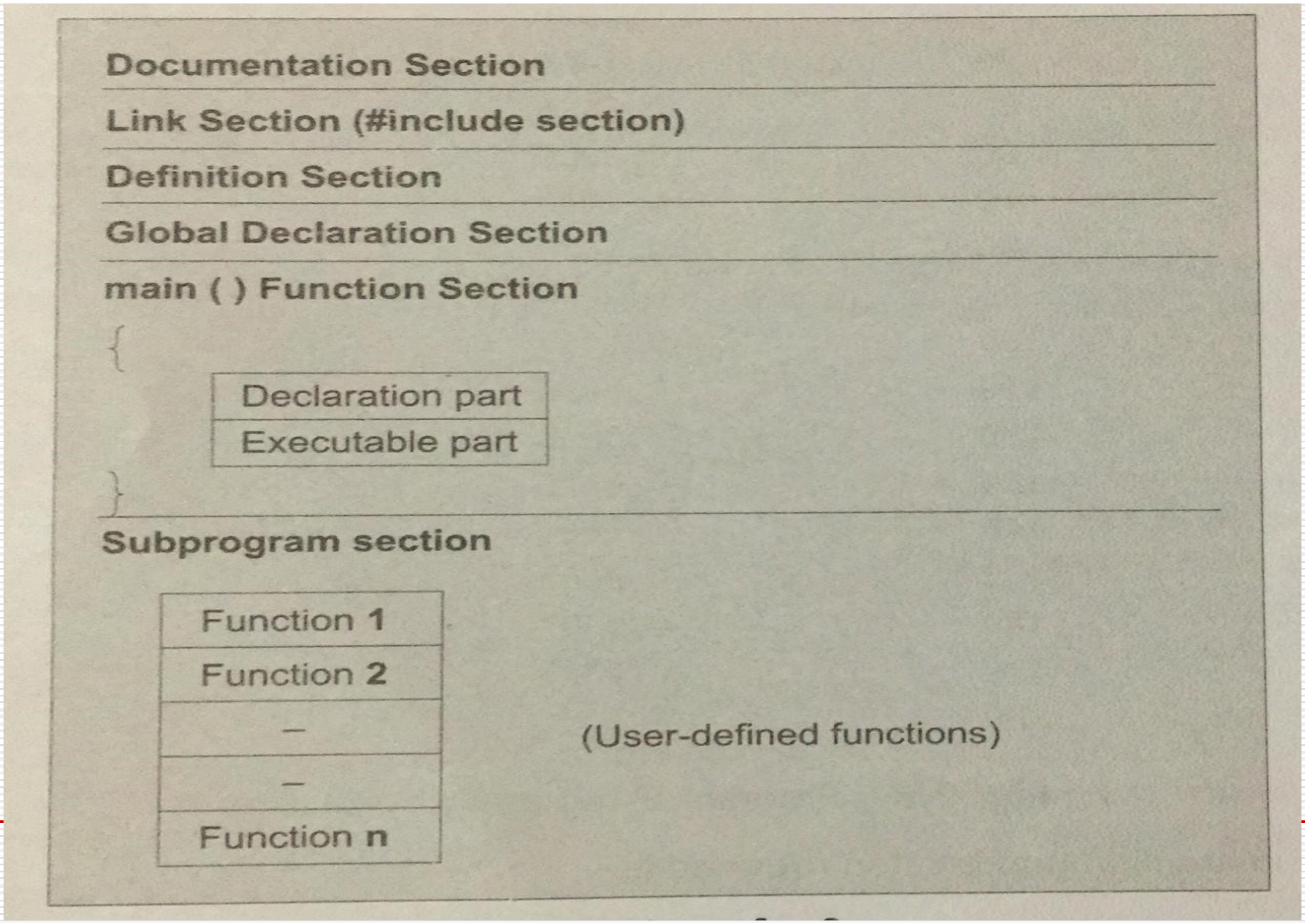


1960 ALGOL International Group
1967 BCPL Martin Richards
1970 B Ken Thompson
1972 Traditional C Dennis Ritchie
1978 K&R C Kernighan and Ritchie
1989 ANSI C ANSI Committee
1990 ANSI/ISO C ISO Committee
1999 C99 Standardization Committee

Facts about ‘C’

- C was invented to write an operating system called UNIX.
- C is a successor of B language which was introduced around 1970
- The language was formalized in 1988 by the American National Standard Institue (ANSI).
- By 1973 UNIX OS almost totally written in C.
- Today C is the most widely used System Programming Language.
- Most of the state of the art software have been implemented using C

Body structure or Program Structure of 'C'



Documentation Section

- It consist set of comments given by the author, programmer or users.
- This section can have the detail about the program. the details can be program definition, programmers name, date, expected output etc…
- Eg.
- //write a program to calculate simple interest

Link Section

- This section provides instructions to the complier to link functions from the system library.
- Here, we can link the header files like stdio, conio, string, math etc…
- Eg.
- #include<stdio.h>

Definition Section

- It contains all the symbolic constants.
- To declare symbolic constant we have to use #define preprocessor directives.
- Eg.
- #define pi 3.14

Global Declaration Section

- There are some variables that are used in more than one function, such variable are known as global variables.
- We can access global variables in main functions as well as subprograms.
- Eg.
- int a =10;

main() function section

- every ‘C’ program must have main() function.
- The actual logic will be put over here.
- **It has two parts:**
  - **Declaration Part**:
    - Here we can declare the local variables.
    - Eg.
    - int a =10;
    - float b=12.2;
  - **Execution Part:**
    - In this part we can perform any logic, calculations etc.
    - Eg.
    - C=a+b;
    - Temp=a;

Subprogram section

- It contains one or more User Defined Functions(UDF).

```
/*
NAME    : ABC XYZ
DATE    : 17/10/2020
PROGRAM : BASIC PROGRAM STRUCTURE */

#include<stdio.h>
#include<conio.h>

#define pi 3.14

int a=10;

void main()
{
        int x=10,y=20,c=0;
        c=x+y;
        printf("%d",c);
        getch();
}

void add()
{
        …
        …

        …
}
```

1.2.2 Character Set, concepts of variables and constants

Character Set

- A set of characters that are allowed to use in computer program is known as Character Set.
- The following character sets are available in ‘C’ programming Language.
  A. Alphabets
  B. Numbers or Digits
  C. Special Characters
  D. White Space Characters

A. Alphabets

- A,B,C,D….Z
- a,b,c,d….z

B. Numbers or Digits

- 0,1,2,3,4….9

C. Special Characters

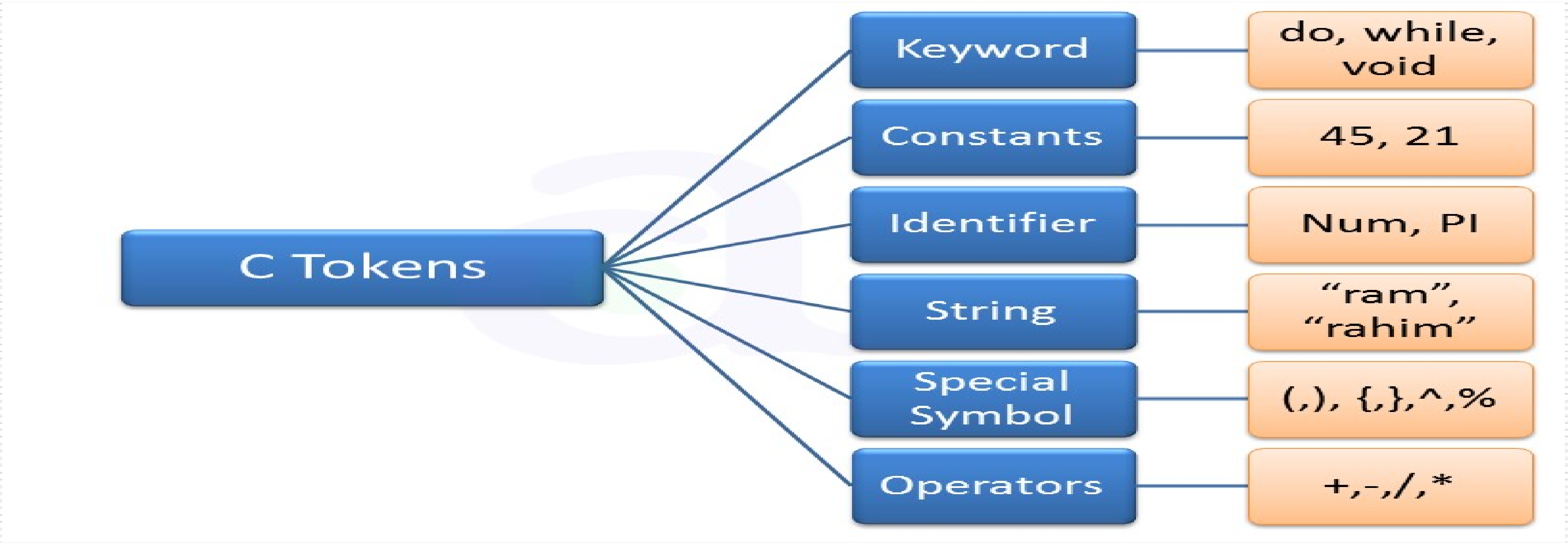
- Minus	+ plus	* asterisk
/ forward slash	\ back slash	(Left parenthesis
) right parenthesis	, comma	; semicolon
: colon	$ dollar sign	. dot operator
> greater than	< less than	= equal to
“ quotation mark	? question mark	! exclamation mark
| vertical bar	~ tilde	_ underscore
^ caret	[opening square bracket	] closing square bracket
{ opening curly bracket	} closing curly bracket	# has sign
@ at symbol	& ampersand	‘ apostrophe
% percentage sign		

D. White Space characters

- Blank Space
- Horizontal Tab
- New Line
- Carriage Return
- Form Feed

1.2.3 Identifiers, literals, Key words

- Token:
  - Smallest individual element or unit 'C' is known as Token.



Keywords:

- Keywords are predefined, reserved words used in programming that have special meanings to the compiler.

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

Identifiers

- Identifiers refers to name of constants, variables, functions, structures.
- **Rules:**
  - First character must be alphabets or underscore.
  - Must be consists letters, digits and underscore.
  - Only 31 characters are significant.
  - Cannot use keyword.
  - Must not contain white space.

Examples:

- stud
- tot
- stud_name
- per

Constants OR Literals

- The fixed values are also called **literals**.
- Constants are those whose values will not change during program executions.
- To declare constant ‘const’ keyword will be used.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
	const int a=10;
	const float pi=3.14;
	clrscr();
	a=20;
	pi=31.4;
	printf("%d",a);
	printf("%f",pi);
	getch();
}
```

Strings:

- Strings are set of characters which can be alphabets, digits or special characters.
- Example:
- “VNSGU”
- “FYBCA- THE Great Class”
- “vnsgu.ac.in”

Special Symbols:

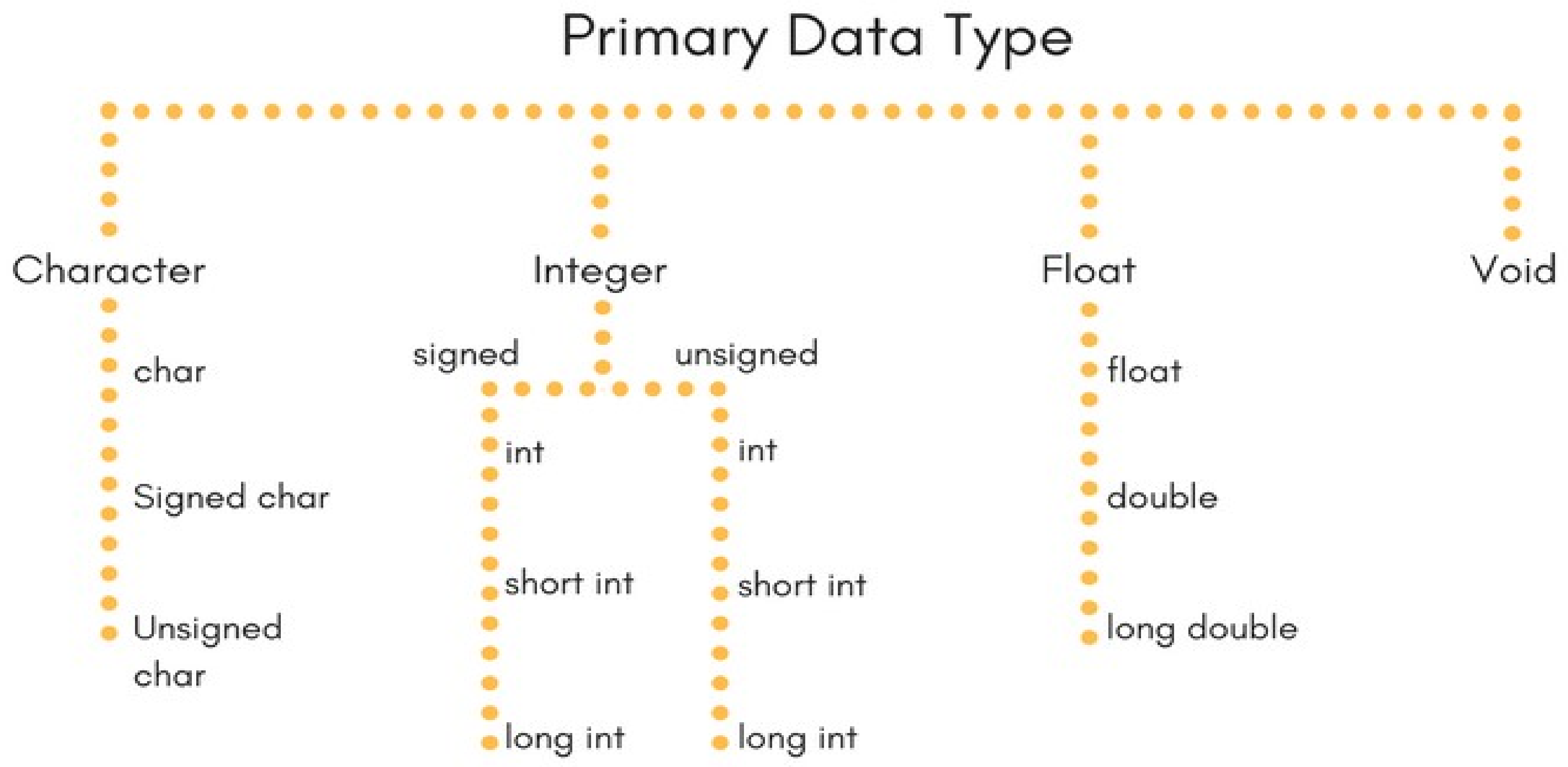
- ‘C’ is having large number of Special symbols or special characters.
- Eg: $,#,@,!,^,& etc…

Operators

- An operators are symbols that tells the compiler to perform specific mathematical or logical functions.
- C language is rich in built-in operators.
  - Arithmetic Operators
  - Relational Operators
  - … etc

1.2.4 Data types (signed and unsigned) (Numeric : int, short int, long, float, double) , (Character type: char, string) and void.

- Data types simply refers to the type and size of data associated with variables.


Primary Data Type
Character
char
Signed char
Unsigned char
Integer
signed
int
short int
long int
unsigned
int
short int
long int
Float
float
double
long double
Void

Integer type

- Integers are used to store whole numbers.

Type	**Size(bytes)**	**Range**	**Format Specifier**
int or signed int	2	-32,768 to 32767	%d
unsigned int	2	0 to 65535	%d
short int or signed short int	1	-128 to 127	%d
unsigned short int	1	0 to 255	%d
long int or signed long int	4	-2,147,483,648 to 2,147,483,647	%ld or %l
unsigned long int	4	0 to 4,294,967,295	%u

Floating point type

- Floating types are used to store real numbers.

Type	Size(bytes)	Range	Format Specifier
float	4	3.4E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	10	3.4E-4932 to 1.1E+4932	%Lf

Character type

- Character types are used to store characters' value.

Type	Size(bytes)	Range	Format Specifier
char or signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c

void type

- void type means no value.
- This is usually used to specify the type of functions which returns nothing.

Data Overflow & Data Underflow

- Assigning a value which is more than its upper limit is called overflow and less than its lower limit is called underflow.

Type Casting OR Type Conversion:

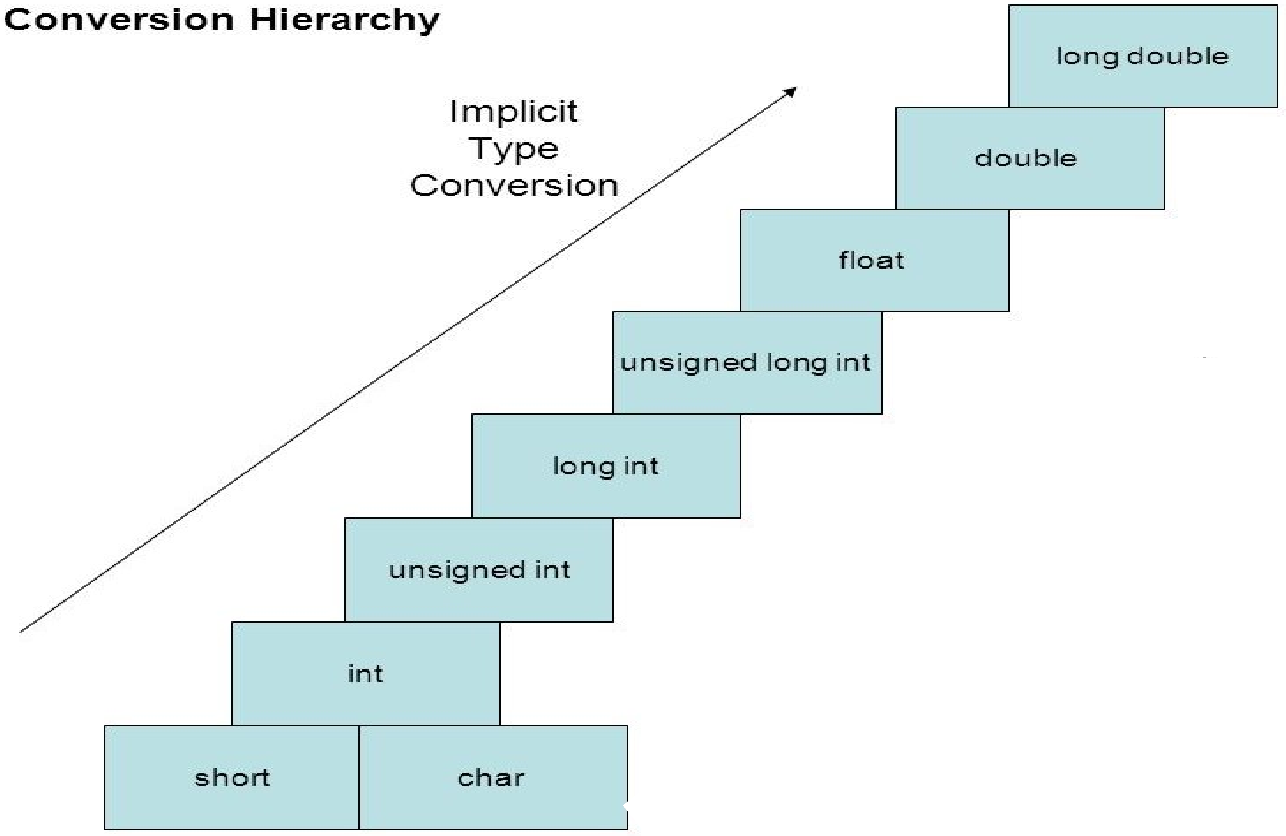
- Type casting is a way to convert a variable from one data type to another data type.
- For example, if you want to store a long value into a simple integer then you can typecast long to int.
- You can convert values from one type to another explicitly using the cast operator.

- There are two forms of Type Casting:
  - Implicit type casting
  - Explicit type casting

Implicit type casting

- It is also known as Automatic Type Casting.
- ‘C’ permits to convert low range data type to high range data types.

Type Conversion Hierarchy



Example:

```
#include<stdio.h>
#include<stdio.h>
void main()
{
	int a =10;
	float b;
	clrscr();
	b=a+100; // Implicit type casting
	printf("%f",b);
	getch();
}
```

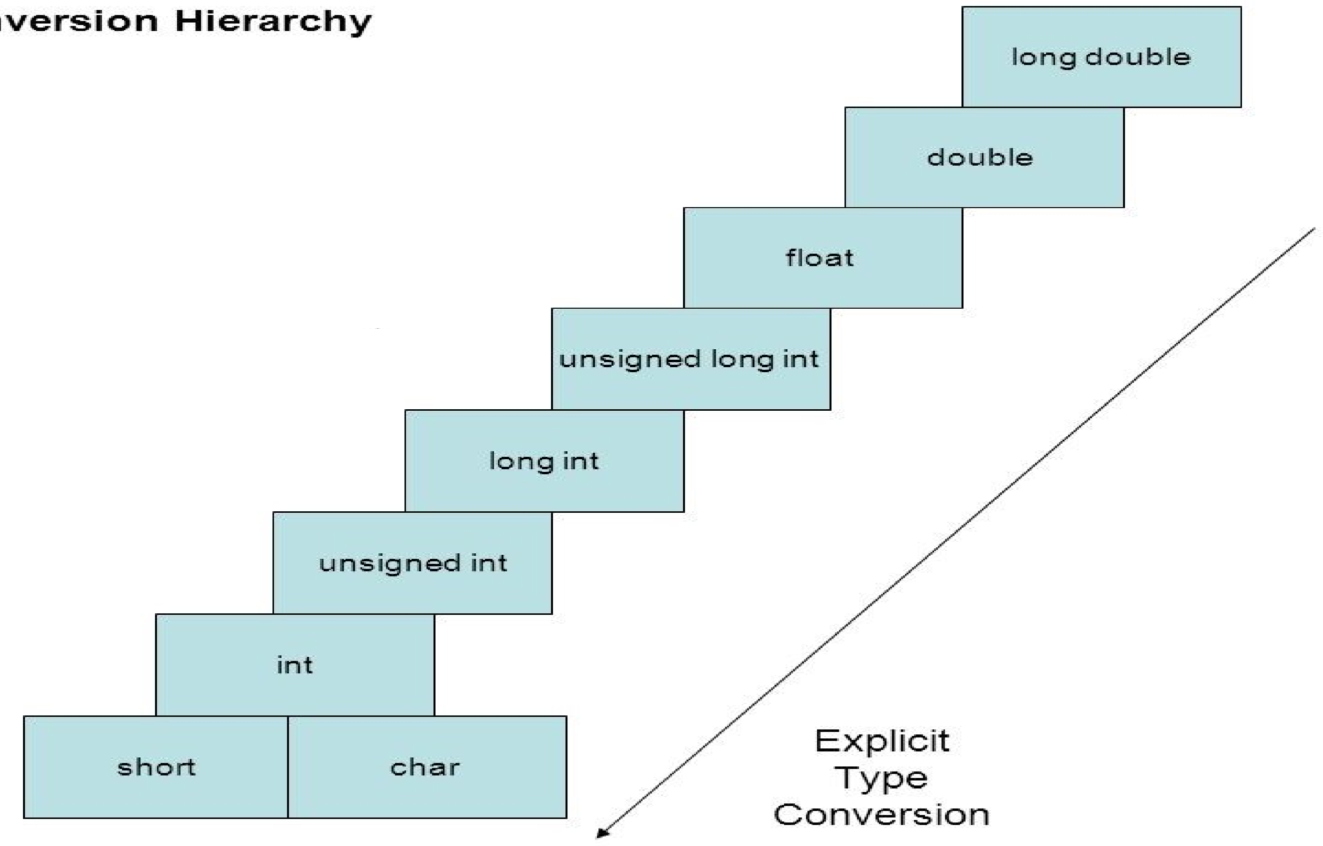
Output:

```
110.000000
```

Explicit type casting

- Explicit Type casting means we force ‘C’ to change the data types.
- When we perform explicit type casting, the variable may loss some value.
- Syntax:
  - (type name) expression;
  - Here type name is any data type like int, float, double…

Type Conversion Hierarchy



Example:

```
#include<stdio.h>
#include<stdio.h>
void main()
{
    int a;
    clrscr();
    a=(int)7.77;// explicit type casting
    printf("%d",a);
    getch();
}
```

Output:

```
7
```

1.2.5 Concepts of source code, object code and executable code

- **Entire user Written Program is known as Source Code.**
- **Source code** is the fundamental component of a computer program that is created by a programmer.
- When a programmer types a sequence of **C** programming language statements into Windows Notepad, for example, and saves the sequence as a text file, the text file is said to contain the **source code.**
- **The Extension of Source code is '.c'**

Object Code:

- **Object code** is the output of a compiler after it processes source **code**.
- **It is Compiled 'C' code.**
- **The Extension of Source code is '.obj'**

Executable Code:

- **Executable (also called the Binary)** is the **output of a linker** after it processes the **object code**. A machine code file can be immediately *executable* (i.e., runnable as a program).
- **The Extension of executable code is '.exe'**.

Comments in ‘C’

- Comments are non-executable statements.
- There are two types of comment in ‘C’.
  - Single Line Comment
  - Multi Line Comments

1. Single Line Comment

- Single line comments are represented by double slash //.
- Example:

```
void main()
{
    //printf("How are You?");
    printf("Hello C");
     getch();
}
```

Output:

```
Hello C
```

2. Multi Line Comment

- Multi line comments are represented by slash asterisk /* ... */.
- It can occupy many lines of code but it can't be nested.
- Syntax:

```
/*
code
to be commented
*/
```

- Example:

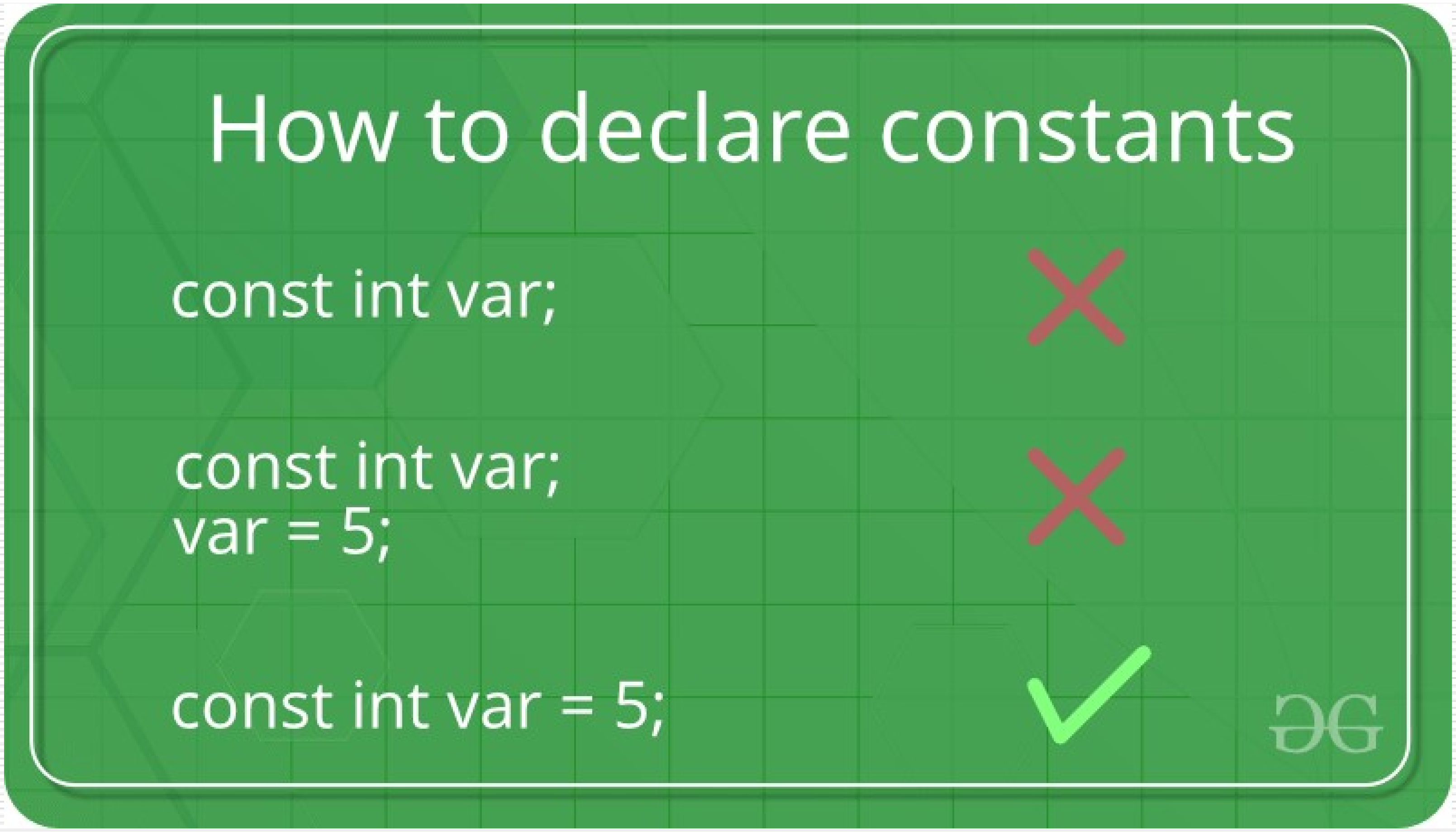
```
#include<stdio.h>
void main()
{
    /*printing information
     Multi Line Comment*/
    printf("Hello C");
}
```

Output:

```
Hello C
```

Constant

- Constants are those whose value will not change during program execution.
- To declare constant ‘const’ will be used.

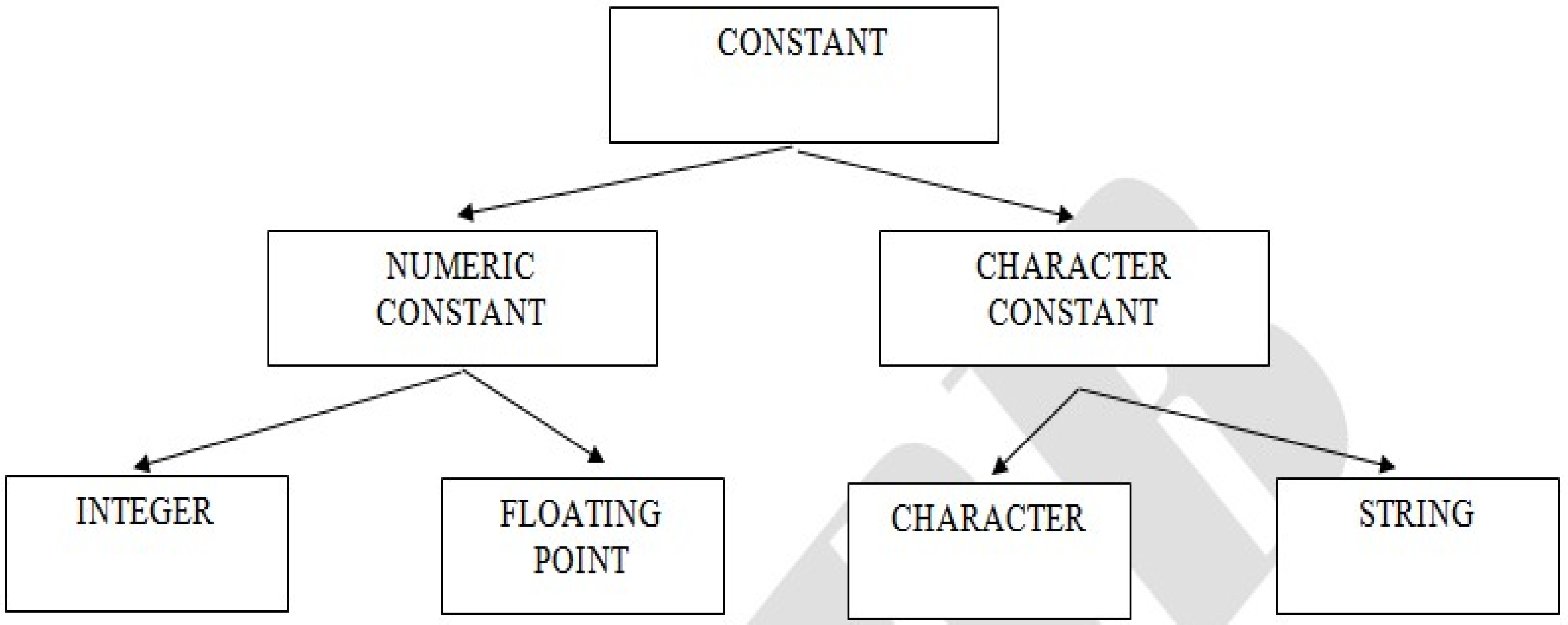


How to declare constants

```
const int var;
```
✗

```
const int var;
var = 5;
```
✗

```
const int var = 5;
```
✓


CONSTANT
NUMERIC CONSTANT
CHARACTER CONSTANT
INTEGER
FLOATING POINT
CHARACTER
STRING

Numeric Constant

- It is used to store numeric data.
- The numeric data can be 1,77,89,88,-99,-23

- There are two types of Numeric Constants:
  - Integer
  - Float

- Integer
    - It refers to the sequence of digits.
    - There are three types of integers
        - Decimal Integer
        - Octal Integer
        - Hexadecimal Integer

- Decimal Integer
  - Consist set of digit from 0 to 9
  - 123,789,-78,98
- Octal Integer
  - consist any number leading with 0(Zero)
  - 012, 011
- Hexadecimal Integer
  - consist any number leading with 0x or 0X

Example:

```
void main()
{
    const int a=10;
    clrscr();
    a=100;
    printf("%d",a);
    getch();
}
```

Output:

It will Give Error (Cannot modify constant object)

Floating Point

- Real constants are used to declare floating point values like: price, percentage, temperature, heights, distance etc…
- Examples: 7.7777, -77.77, 0.0005, 1.5e2

Example:

```
void main()
{
    const float pi=3.14;
    clrscr();
    pi=31.4;
    printf("%f",pi);
    getch();
}
```

Output:

It will Give Error (Cannot modify constant object)

Character Constant

- There are two category of Character Constants:
  - Character
  - String

- **Character**:
- Character constant represents a single character within a single quote(‘ ‘).
- Example: ‘A’, ‘Z’, ‘*’, ‘ ‘, ‘5’

Example:

```
void main()
{
    const char a='c';
    clrscr();
    a='z';
    printf("%c",a);
    getch();
}
```

Output:

```
z
```

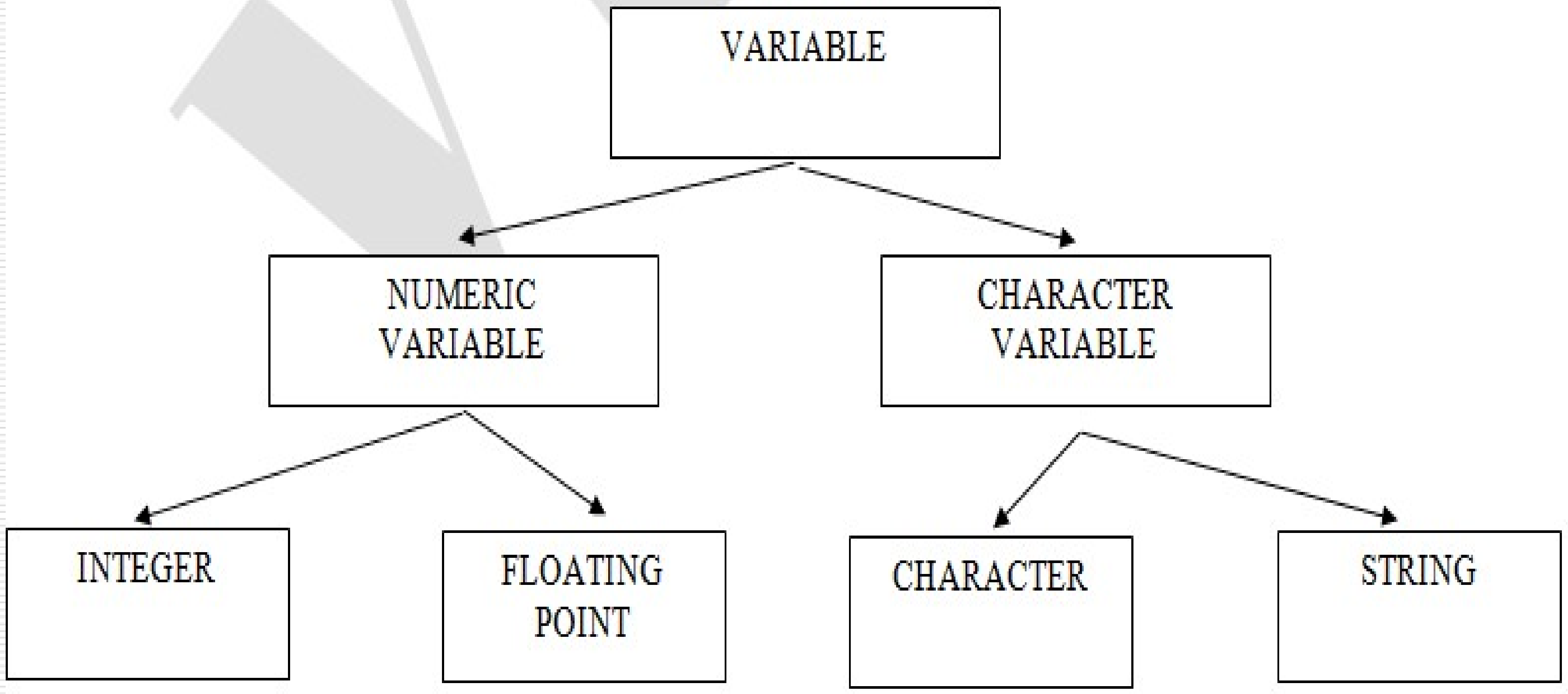
- **String**:
- A string constant is a sequence of character enclosed in “ “ (double quote).
- Example: “BCA”, ”VNSGU”, “BBA”, “5+3”.

Example:

```
void main()
{
    const char a[50]="VNSGU";
    clrscr();
    printf("%c",a);
    getch();
}
```

Variable:

- Variables are those whose value will be change during program execution.


VARIABLE
NUMERIC VARIABLE
CHARACTER VARIABLE
INTEGER
FLOATING POINT
CHARACTER
STRING

Numeric Variable

- It is used to store numeric data.
- The numeric data can be 1,77,89,88,-99,-23
- There are two types of Numeric Constants:
  - Integer
  - Float

- Integer
  - It refers to the sequence of digits.
  - There are three types of integers
    - Decimal Integer
    - Octal Integer
    - Hexadecimal Integer

- Decimal Integer
  - Consist set of digit from 0 to 9
  - 123,789,-78,98
- Octal Integer
  - consist any number leading with 0(Zero)
  - 012, 011
- Hexadecimal Integer
  - consist any number leading with 0x or 0X

Example:

```
void main()
{
    int a=10;
    clrscr();
    a=100;
    printf("%d",a);
    getch();
}
```

Output:

```
100
```

- Floating Point
  - Real constants are used to declare floating point values like: price, percentage, temperature, heights, distance etc…
  - Examples: 7.7777, -77.77, 0.0005, 1.5e2

Example:

```
void main()
{
    float pi=3.14;
    clrscr();
    pi=31.4;
    printf("%f",pi);
    getch();
}
```

Output:

```
31.400000
```

Character Constant

- There are two category of Character Constants:
  - Character
  - String

- **Character**:
- Character constant represents a single character within a single quote(‘ ‘).
- Example: ‘A’, ‘Z’, ‘*’, ‘ ‘, ‘5’

Example:

```
void main()
{
    char a='c';
    clrscr();
    a='z';
    printf("%c",a);
    getch();
}
```

Output:

```
z
```

- **String**:
- A string constant is a sequence of character enclosed in “ “ (double quote).
- Example: “BCA”, ”VNSGU”, “BBA”, “5+3”.

Example:

```
void main()
{
    const char a[50]="VNSGU";
    clrscr();
    printf("%s",a);
    getch();
}
```