Programming for Problem Solving

(ESCG101)

Course Details

- Program Name: B.Tech (CSE)
- Course Name: Programming for Problem Solving
- Course Code: ESCG101
- Contact: 3L
- Credit: 3
- Pre-requisite
  - basic concept of algorithm
  - basic concept of flowchart

Complete Syllabus

- **Module 1** - Introduction to Programming: Introduction to components of a computer system (disks, memory, processor, operating system, compilers). Idea of Algorithm: steps to solve logical and numerical problems. Representation of Algorithm. Flowchart/Pseudocode with examples. Algorithms to programs; source code, variables (with data types) variables and memory locations, Syntax and Logical Errors in compilation, object and executable code.
- **Module 2 -** Arithmetic expressions and precedence ,Conditional Branching and Loops: Writing and evaluation of conditionals and consequent branching ,Iteration and loops, Arrays: Arrays (1-D, 2-D), Character arrays and Strings
- **Module 3 -** Function: Built in libraries, Parameter passing in functions, call by value, Passing arrays to functions: call by reference, Recursion: Example programs, Finding Factorial, Fibonacci series, Ackerman function, Quick sort or Merge sort.
- **Module 4 -** Structure: Structures, Defining structures and Array of Structures, Pointers: Idea of pointers, Defining pointers, Use of Pointers in self-referential structures, notion of linked list
- **Module 5 -** File handling, File concepts, parameters, modes of operation

Textbooks/References

- Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill
- E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill
- Gary J. Bronson, A First Book of ANSI C, 4th Edition, ACM
- Kenneth A. Reek, Pointers on C, Pearson
- Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India

Course Objectives

From this course, the students will able to:

- Define the basic concepts of computer system and recall the ideas of algorithm and flowchart
- Discuss the basic concepts and ideas of C programming language
- Explain the concepts of loops and functions and apply in solving a problem
- Identify the use of structure and pointer and compare the importance of using file in programming language

Module 1

Contents of Module 1

- Introduction to Programming: Introduction to components of a computer system (disks, memory, processor, operating system, compilers).
- Idea of Algorithm: steps to solve logical and numerical problems. Representation of Algorithm.
- Flowchart/Pseudocode with examples. Algorithms to programs; source code, variables (with data types) variables and memory locations, Syntax and Logical Errors in compilation, object and executable code

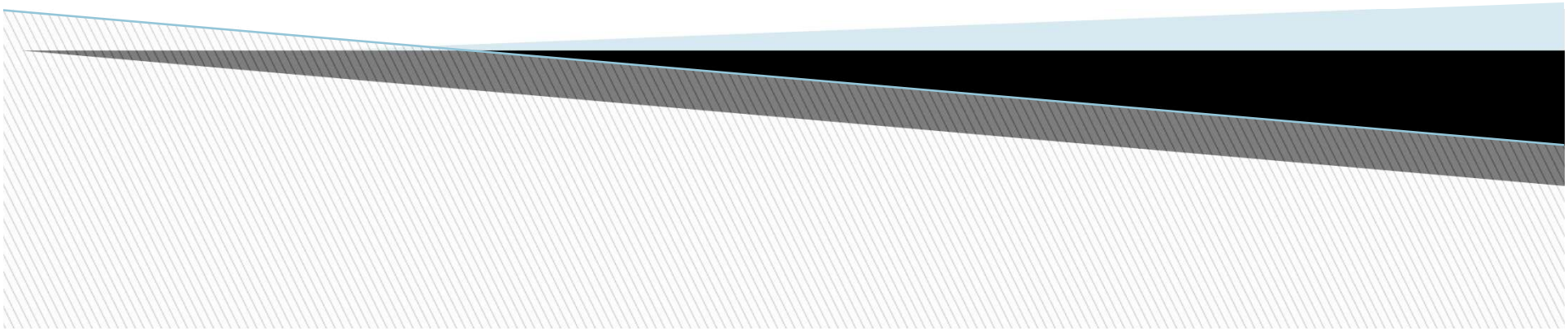
Objectives of Module 1

- Learn components of a computer system like disks, memory, processor, operating system, compilers etc.
- Get the idea of algorithm to solve problems.
- Draw flowchart for a problem.
- Learn different data types, variables and memory locations.
- Identify errors like syntax and logical errors.
- Learn different types of operators.

Introduction to Programming

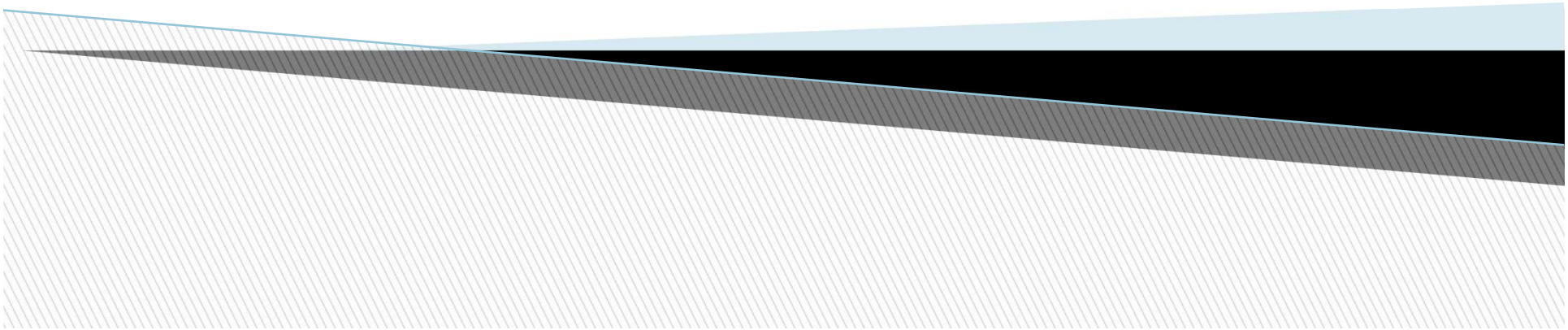
Programming Languages: Programming languages are used to communicate instructions to computers.

Examples include C, Python, Java and many more. Each language has its syntax, rules, and purposes.



CONTD..

- **Syntax:** refers to the rules governing the structure of code in a programming language. Correct syntax is essential for the code to run without errors.
- **Debugging:** is the process of identifying and fixing errors (bugs) in code.



Computer System

- Input devices
- CPU (CU+ALU+MU)
- Output devices

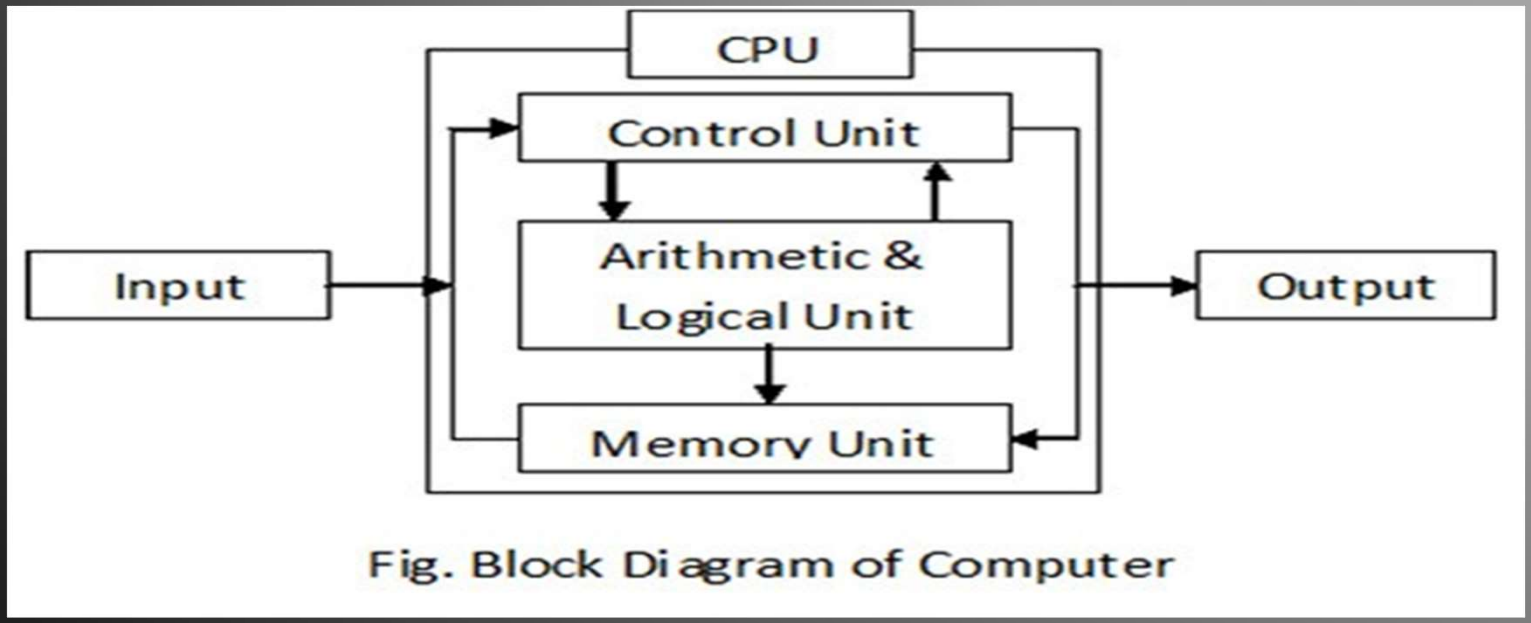


Fig. Block Diagram of Computer

Computer System

- Example of input device – keyboard, mouse etc
- Example of output device – monitor, printer etc
- Control unit (CU) is the brain of computer
- Arithmetic Logic unit (ALU) is used for performing arithmetic and logical operations
- Memory unit (MU) is the memory used for storing data
  - Primary memory, secondary memory

Algorithm

- It is a step by step method of solving a problem
- Example: to add two numbers

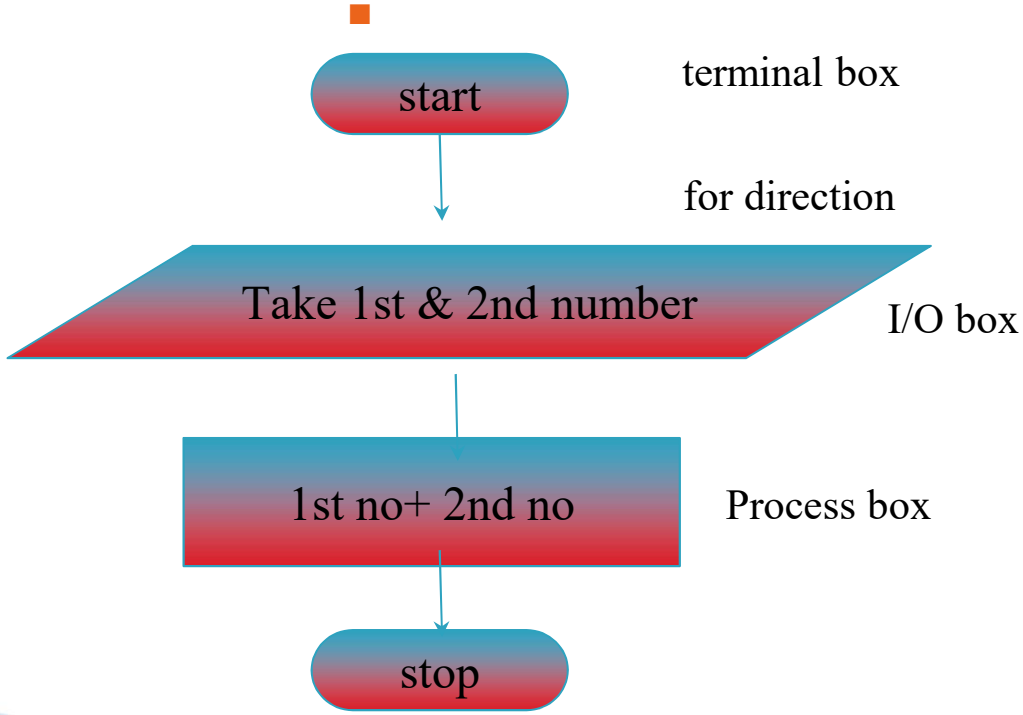
1. start
2. take first number
3. take second number
4. add first and second number
5. stop

Features of an algorithm

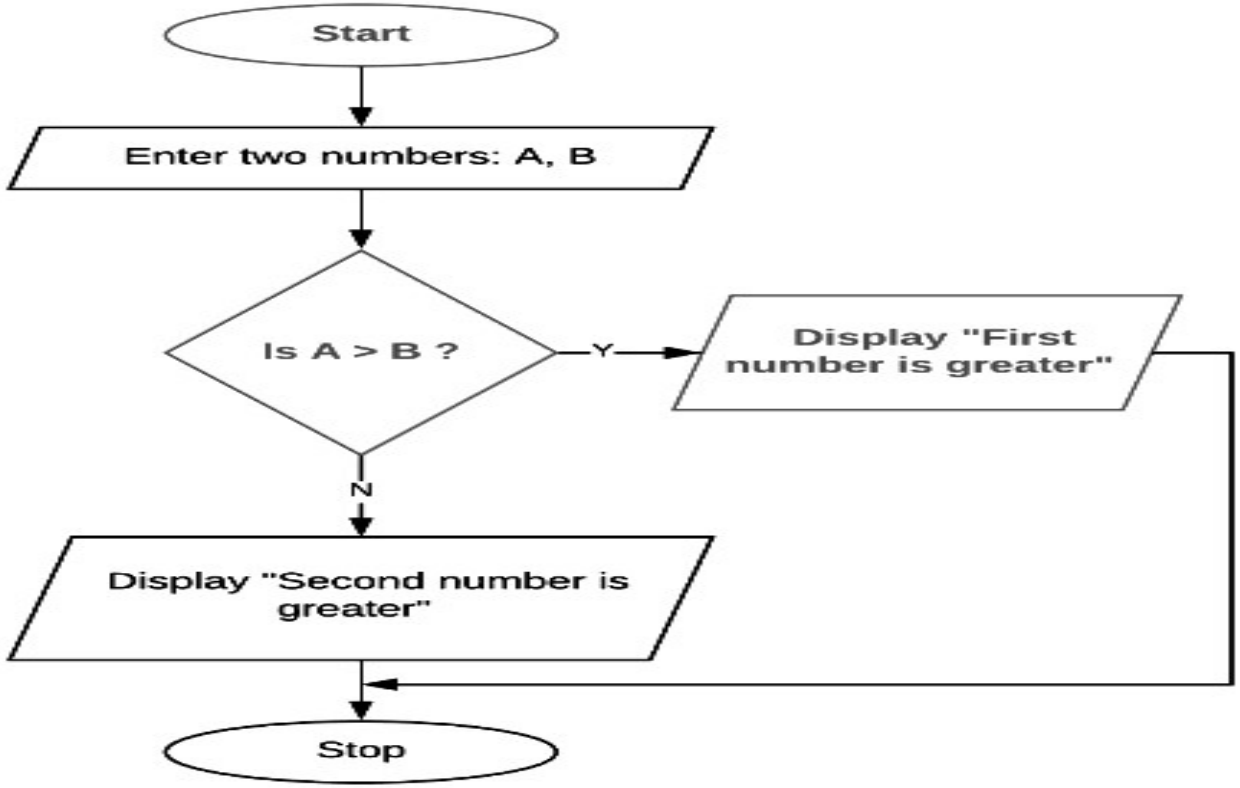
- Precision
  - steps precisely defined
- Uniqueness
  - no ambiguity
- Finiteness
  - end to some solution
- Input/Output

Flowchart

- The pictorial representation of the step by step solution of a given problem
- Pictorial representation of an algorithm



Flowchart for greatest of two numbers



Error and Types

- Unexpected happening or illegal operation performed by the user which results in abnormal working of the program
- Types:
  - Syntax-- violates the rules (compile time)
  - Logical-- unexpected due to logic
    - writing a+b rather than a-b
  - Run-time-- say (a/b), it works except b=0

C Programming Language

- Developed by Dennis Ritchie
- In 1972 at AT&T Bell Laboratory
- Keyword- word with predefined meaning
  - Ex-- int, if, void etc
- Identifier- any name given to a variable, function, array etc
- Variable- a name to store data in memory location

Classifying Data types

- Basic (primitive/primary) types include:
  - Integers
  - Characters
  - Floating points (real numbers)
  - Boolean
- Secondary/derived types include:
  - Array
  - Pointer
  - Structure
  - String

Uses

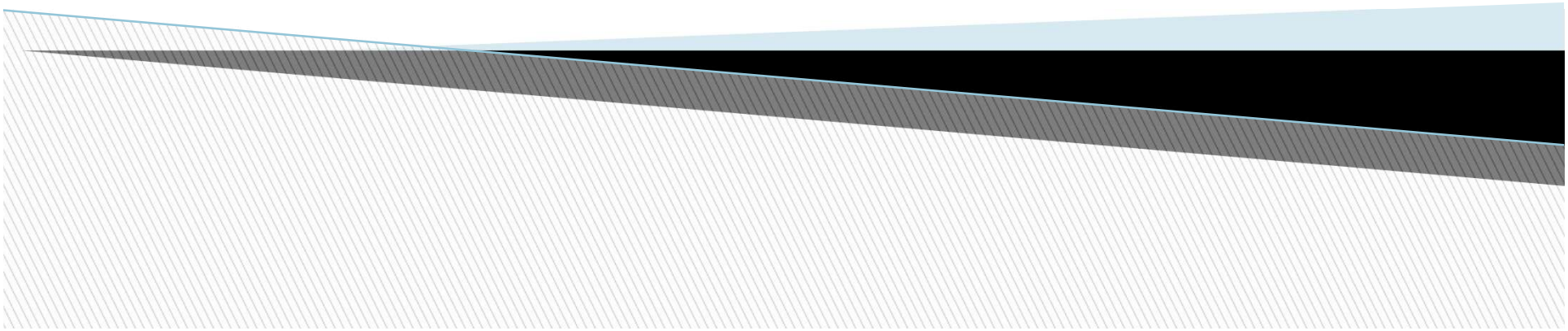
- 'int' (Integers) is used for storing integers values (-5, 0, 5 etc.)
- 'float' is used to for storing decimal numbers (5.5, -5.7 etc.)
- 'char' (character) is used for storing single letter or special symbols (a,C,$ etc.)

Storage Capacity

- 'int' = 2 bytes (16 bit compiler)
  = 4 bytes (32 bit compiler)
- 'float' = 4 bytes
- 'char' = 1 byte

Calculate the total memory required (both bytes and kilobytes) to store an array of 156 integers and 52 floating-point numbers where the machine has 32bit compiler?

Ans: 832 bytes (0.8125KB)



Format specifier

- Format specifiers are needed to access different data types
- int - %d
- float - %f
- char - %c
- String - %s
- Pointer- %p

Keywords

- They are the fixed defined words with some meaning
- Example :
  - **Void:** is a keyword in C used to indicate that a function does not return any value. It is typically used in function declarations and definitions to specify that the function does not produce a result that can be used by other parts of the program.
  - **main:** is required in every C program, and it must return an integer value (usually 0 to indicate successful execution) to the operating system when the program terminates.

Identifier

- Any name assigned to represent something
- Example :
  - Say roll number 1 represents a student
  - Similarly any name used to represent in C language like variable, function etc

Variable

- Used to hold values
- It is a named-storage location that holds a value.
- Example : int var;
  - Here var is a variable that will hold integer data type values
  - There can be any name except keywords

Rules for Identifier Name

- Any name except keywords like int, float, if etc.
- It can contains alphabets, numbers etc.
- Must start with a letter
- Contains special character
  - Underscore (_) only
- First character of name cannot be a number
- It is case-sensitive

Operators

- An operator is a symbol that represents a specific operation to be performed on one or more operands.
- Operator is used to manipulate data and variables
- Example: a + b, where a & b are operands and + is an operator

Types of Operators

- Arithmetic Operators
- Relational Operators
- Assignment Operator
- Logical Operators
- Conditional Operators
- Bitwise Operators
- Increment/Decrement Operators

Arithmetic Operators

- Arithmetic operators used to perform arithmetic operations
- The arithmetic operators in C language:

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo

Relational Operators

- Also known as comparison operators, are symbols used in programming languages to compare two values or expressions. These operators allow you to determine the relationship between the values and make decisions based on their comparisons. The relational operators are:

>	Greater than
<	Less than
>=	greater than Equal to
<=	Less than Equal to
==	Equal to
!=	Not Equal to

Assignment Operators

- Assignment operators are used to assign value to a variable
- '= ' sign is used to assign a value to variable
- Example: int a = 8;

Logical Operators

- Logical operators are used in programming to perform logical operations on boolean values (true or false).

Examples:

- Logical AND (&&): It returns ‘true’ if both of it’s operands are true, otherwise it returns false.
- Logical OR (||): logical OR operator returns true if at least one of its operands is true, and it returns false if both operands are false.
- Logical NOT (!): The logical NOT operator negates the value of its operand. If the operand is true, it returns false, and if the operand is false, it returns true.

Conditional Operators

The conditional operator, often referred to as the ternary operator, is a concise way to create conditional expressions. It's used to make decisions between two values or expressions based on a condition.

Syntax: condition ? value_if_true : value_if_false

- value_if_true: This is the value or expression that is returned if the condition is true.

- value_if_false: This is the value or expression that is returned if the condition is false.

CONTD..

Example:

```
#include <stdio.h>

int main() {
    int x = 10;
    int y = 20;
        int max_value = (x > y) ? x : y;
    printf("The maximum value is: %d\n",max_value);
 return 0;
}
```

Bitwise Operators

- Bitwise operators in C are used to perform operations at the bit level of integers. They manipulate individual bits of an integer value rather than the entire value itself.
- There are various bitwise operators in C as following:

&	Bitwise AND
|	Bitwise OR
<<	Left Shift
>>	Right Shift

CONTD..

Bitwise AND (&):

Performs a bitwise AND operation between the corresponding bits of two operands.

Eg: A=5

B=3 A & B=?

ANS. 1

Truth Table of Bitwise AND (&) Operator

X	Y	X & Y
0	0	0
0	1	0
1	0	0
1	1	1

CONTD..

Bitwise OR (|):

Performs a bitwise OR operation between the corresponding bits of two operands.

Eg: A=5

B=3 A | B=?

ANS. 7

Truth table

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

CONTD..

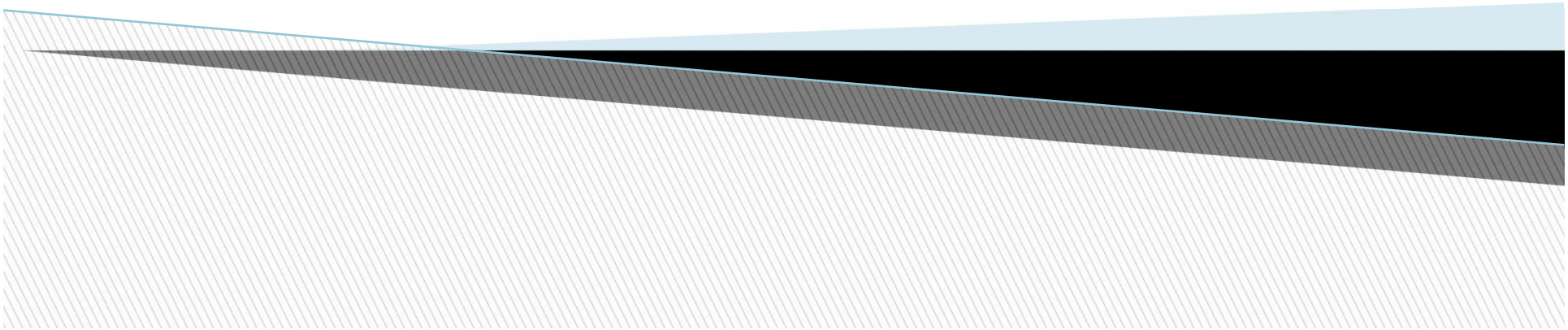
Left Shift (<<):

Shifts the bits of the left operand to the left by a specified number of positions.

Eg: A=5

A<<2=?

ANS. 20



CONTD..

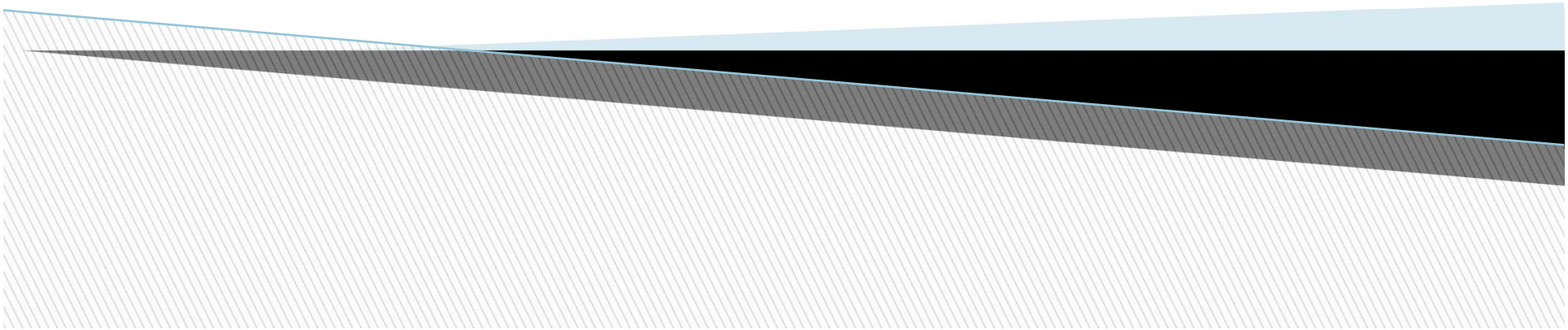
Right Shift (>>):

Shifts the bits of the left operand to the right by a specified number of positions.

Eg: A=15

A>>2=?

ANS. 3



Questions:

1) A=12
   B=25 A & B=?
   Ans. 8

2) A=25
   B=18 A | B=?
   Ans. 27

3) A=13 A>>2=? Ans. 3

4) A=25 25<<2=?
   Ans. 100

Increment/Decrement Operators

- Increment operator are used to increase the value by 1
- Decrement operator decrease the value by 1
- Increment operator is denoted by ++
- Decrement operator is denoted by --

Increment/Decrement Operators

Increment:
int x=5;
x++; //the value of x become 5+1=6

Decrement:
int x=5;
x-- ; //the value of x
become 5-1=4

Precedence of Operators

- Operator precedence is a crucial concept in programming that determines the order in which operators are evaluated in expressions.
- Like out of three operators like *, / and +, which is executed first?
- Precedence in decreasing order: (++, --), (*, /, %), (+, -), (<<, >>), (<, <=, >, >=), (==, !=), &, |, &&, ||.
- Operators in bracket are having same priority

CONTD..

Example:

1) Int a=5, b=3, c=7;

a+b*c/a- c%b=? 8

2) x=5, y=10, z=15;

x+y*z-y/x =? 153

CONTD..

3) X=5, Y=2, Z=8;

X>Y&&Y<Z= ? 1(True)

4) a=5, b=2, c=8;

++a*b-- +c=? 20

5) a=3, b=4, c=5;

++a ||b-- && c++ =? 1 (True)

Simple C program

```
void main()
{
   printf("Hello World");
}

// here main is a library function where execution starts
//printf is a function to display on screen (here, Hello
   World)
// semi colon is used to terminate a statement
```

Simple C program

```
void main()
{
   int num1=5, num2 =2;
   int add;
   add = num1 + num2;
   printf("Addition is %d",add);
}
//num1, num2 and add are integer variables
```

Storage Class

- In the C programming language, a storage class defines the scope (visibility) and lifetime of a variable or a function within a program. C provides several storage classes that determine how variables and functions are stored in memory and how they can be accessed by different parts of a program.

Types of Storage class

- Automatic (auto) -- local variable & default garbage value
- Register -- local variable stored in register and default garbage value
- Static -- local variable & with 0 default value
- External (extern) -- global variable & 0 default value