

# Unit : 1

## Introduction

# Introduction


- C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. C was originally first implemented on the DEC PDP-11 computer in 1972.
- In 1978, Brian Kernighan and Dennis Ritchie produced the first publicly available description of C, now known as the K&R standard.

- the C compiler, and essentially all UNIX application programs have been written in C. C has now become a widely used professional language for various reasons.

- ❖ Easy to learn
- ❖ Structured language
- ❖ It produces efficient programs
- ❖ It can handle low-level activities
- ❖ It can be compiled on a variety of computer platforms

# Facts about C

- C was invented to write an operating system called UNIX.
- C is a successor of B language which was introduced around the early 1970s.
- The language was formalized in 1988 by the American National Standard Institute (ANSI).
- The UNIX OS was totally written in C.
- Today C is the most widely used and popular System Programming Language.

- 
- Most of the state-of-the-art software have been implemented using C.
  - Today's most popular Linux OS and RDBMS MySQL have been written in C.

# Why use C?

- C was initially used for system development work, particularly the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as the code written in assembly language. Some examples of the use of C might be.

- 
- Operating Systems
  - Language Compilers
  - Assemblers
  - Text Editors
  - Print Spoolers
  - Network Drivers
  - Modern Programs
  - Databases
  - Language Interpreters
  - Utilities

# C Programs

- A C program can vary from 3 lines to millions of lines and it should be written into one or more text files with extension ".c".



# Text Editor

- This will be used to type your program. Examples of few a editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.
- The name and version of text editors can vary on different operating systems. For example, Notepad will be used on Windows, and vim or vi can be used on windows as well as on Linux or UNIX.

- The files you create with your editor are called the source files and they contain the program source codes. The source files for C programs are typically named with the extension ".c".

# The C Compiler

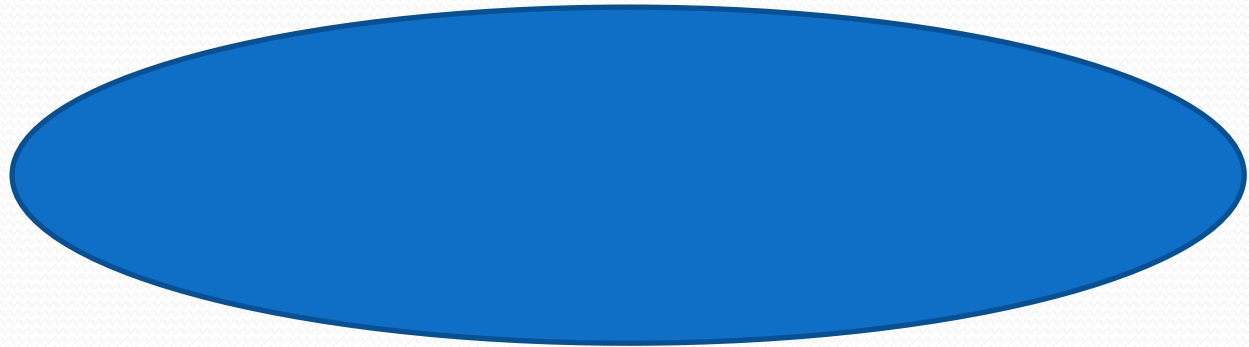
- The source code written in source file is the human readable source for your program. It needs to be "compiled", into machine language so that your CPU can actually execute the program as per the instructions given.
- The compiler compiles the source codes into final executable programs. The most frequently used and free available compiler is the GNU C/C++ compiler,

# Basic programming techniques

- A **flowchart** is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem.
- Flow chart use various symbols to define state of program.

# Start / Stop

- Used to represent start and end of flowchart.



# Input / Output

- Used for input and output operation.



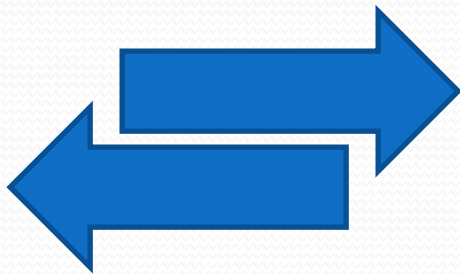
# Process Box

- Used for arithmetic operations and data-manipulations.



# Lines / Flow

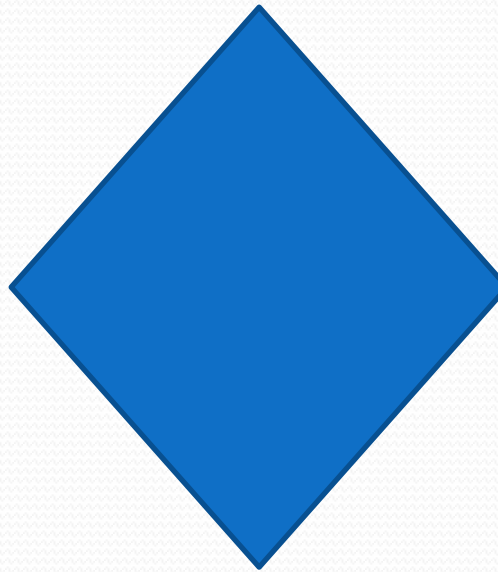
- Used to indicate the flow of logic by connecting symbols.





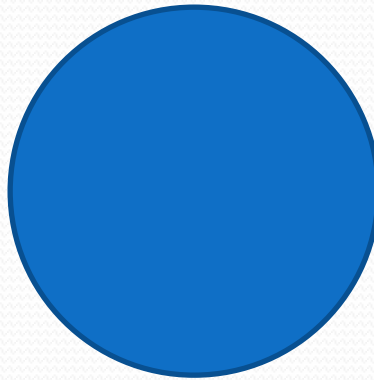
# Condition

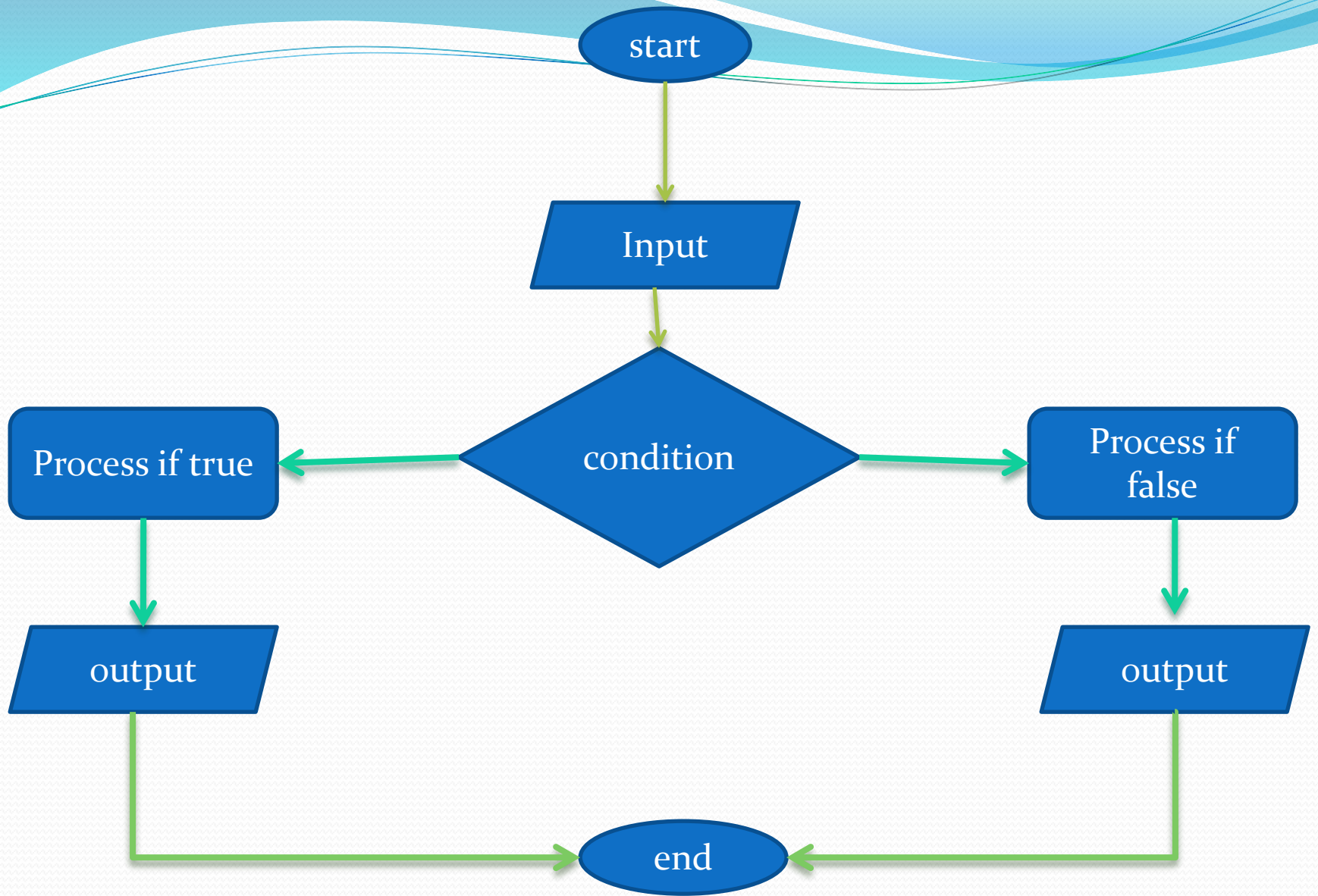
- Used to represent the operation in which there are two alternatives, true and false.



# Connector

- Used to join different flowline





# C Program Structure

- A C program basically consists of the following parts –
  - ❖ Preprocessor Commands
  - ❖ Functions
  - ❖ Variables
  - ❖ Statements & Expressions
  - ❖ Comments

# Simple Program

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    /* my first program in C */
    printf("Hello, World! \n");
    getch();
}
```

# Explanation

- The first line of the program *#include <stdio.h>* is a preprocessor command, which tells a C compiler to include `stdio.h` file before going to actual compilation.
- The next line *int main()* is the main function where the program execution begins.

- The next line `/*...*/` will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
- The next line `printf(...)` is another function available in C which causes the message "Hello, World!" to be displayed on the screen.

# Compile and Execute C Program

- Use alt + F9 for Compile Your Source code.
- Use ctrl + f9 to run your program.



# Must Keep in Mind.

- C Language is totally case sensitive language.
- **A** and **a** both are different for c compiler.

# Tokens in C Language.

- A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following C statement consists of five tokens .

# Semicolons

- In a C program, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

# Comments

- Comments are like helping text in your C program and they are ignored by the compiler. They start with `/*` and terminate with the characters `*/` as shown below.
- `//` for single line comment
- `/*` for multiline comment `*/`

# Identifiers

- A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z, a to z, or an underscore '\_' followed by zero or more letters, underscores, and digits (0 to 9).

- C does not allow punctuation characters such as @, \$, and % within identifiers. C is a **case-sensitive** programming language. Thus, *Manpower* and *manpower* are two different identifiers in C. Here are some examples of acceptable identifiers –

# Valid / invalid variable names

## Valid

- a1
- A
- AA
- Abc
- Abc\_d
- \_d
- Rajkot
- My\_var
- val1

## invalid

- 1a
- #a
- \$abc
- val@1
- Value-1
- V\*
- 1val
- Int
- char

# Keywords

- The following list shows the reserved words in C. These reserved words may not be used as constants or variables or any other identifier names. The reserved keyword have some specific meaning in C compiler.



# List of Keywords

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

# Whitespace in C

- A line containing only whitespace, possibly with a comment, is known as a blank line, and a C compiler totally ignores it.
- Whitespace is the term used in C to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as `int`, ends and the next element begins.

# Back ground files

- When you save and compile you can find following files in your bin folder of TC.

file type

extension

C- Source file

.c

application file

.exe

object file

.obj

backup file

.bak

# Data Types

- Data types in c refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

# Basic Types

- They are arithmetic types and are further classified into: (a) integer types and (b) floating-point types.

# Enumerated types

- They are again arithmetic types and they are used to define variables that can only assign certain discrete integer values throughout the program.

# The type `void`

- The type specifier *void* indicates that no value is available.

# Derived types

- They include (a) Pointer types, (b) Array types, (c) Structure types, (d) Union types and (e) Function types.
- The array types and structure types are referred collectively as the aggregate types. The type of a function specifies the type of the function's return value.



# Integer Types

- Signed Integer
  - Memory Occupied 2 Bytes.
  - Total bits 16
  - Total usable bits 15
  - First bit is occupied for sign bit ( + / -)
  - range -32,768 to 32,767

# Integer Types

- unsigned Integer
  - Memory Occupied 2 Bytes.
  - Total bits 16
  - Total usable bits 16
  - Store only positive values.
  - range 0 to 65535

Note : to create unsigned integer variable must use unsigned before int keyword.

# Floating-Point Types

Type	Storage Size	Value Range	Precision
float	4 byte	$1.2\text{E-}38$ to $3.4\text{E+}38$	6 decimal places
double	8 byte	$2.3\text{E-}308$ to $1.7\text{E+}308$	15 decimal places
long double	10 byte	$3.4\text{E-}4932$ to $1.1\text{E+}4932$	19 decimal places

# char

- Use to store single character.
  - Memory Occupied 1 Bytes.
  - Total bits 8
  - Total usable bits 8
  - range 0 to 255

# ASCII

- American Standard Code for Information Interchange, is a character encoding standard (the Internet Assigned Numbers Authority (IANA) prefers the name US-ASCII). **ASCII** codes represent text in computers.

# Sizeof Operator


- To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** operator. The expressions *sizeof(type)* yields the storage size of the object or type in bytes.

# Variables

- A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

- The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive. Based on the basic types explained in the previous chapter, there will be the following basic variable types.





• Type	Description
char	Typically a single octet(one byte). This is an integer type.
int	The most natural size of integer for the machine.
float	A single-precision floating point value.
double	A double-precision floating point value.

- C programming language also allows to define various other types of variables, which we will cover in subsequent chapters like Enumeration, Pointer, Array, Structure, Union, etc. For this chapter, let us study only basic variable types.

# Variable Definition in C

- A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type and contains a list of one or more variables of that type as follows.
- `type variable_list;`

- Here, **type** must be a valid C data type including char, w\_char, int, float, double, bool, or any user-defined object; and **variable\_list** may consist of one or more identifier names separated by commas. Some valid declarations are shown here.
- int i, j, k;
- char c, ch;
- float f, salary;
- double d;

- The line **int i, j, k;** declares and defines the variables i, j, and k; which instruct the compiler to create variables named i, j and k of type int.

# Variable initialized

- Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows.
- `type variable_name = value;`
- `int d = 3, f = 5; // definition and initializing d and f.`
- Note : newly created variable have garbage values.

# Variable Declaration in C

- A variable declaration provides assurance to the compiler that there exists a variable with the given type and name so that the compiler can proceed for further compilation without requiring the complete detail about the variable. A variable definition has its meaning at the time of compilation only, the compiler needs actual variable definition at the time of linking the program.

# Lvalues and Rvalues in C

- **lvalue** – Expressions that refer to a memory location are called "lvalue" expressions. An lvalue may appear as either the left-hand or right-hand side of an assignment.
- **rvalue** – The term rvalue refers to a data value that is stored at some address in memory. An rvalue is an expression that cannot have a value assigned to it which means an rvalue may appear on the right-hand side but not on the left-hand side of an assignment.



# Example

- `int g = 20; // valid statement`
- `10 = 20; // invalid statement; would generate compile-time error`

# C - Constants & Literals

- Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called literals.
- Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal. There are enumeration constants as well.
- Constants are treated just like regular variables except that their values cannot be modified after their definition.

# Integer Literals

- An integer literal can be a decimal, octal, or hexadecimal constant. A prefix specifies the base or radix: `0x` or `0X` for hexadecimal, `0` for octal, and nothing for decimal.
- An integer literal can also have a suffix that is a combination of `U` and `L`, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.

- 212            /\* Legal \*/
- 215u        /\* Legal \*/
- 0xFeeL     /\* Legal \*/
- 078         /\* Illegal: 8 is not an octal digit \*/
- 032UU      /\* Illegal: cannot repeat a suffix \*/

# Floating-point Literals

- A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. You can represent floating point literals either in decimal form or exponential form.
- While representing decimal form, you must include the decimal point, the exponent, or both; and while representing exponential form, you must include the integer part, the fractional part, or both. The signed exponent is introduced by e or E.

- 3.14159            /\* Legal \*/
- 314159E-5L       /\* Legal \*/

# Character Constants

- Character literals are enclosed in single quotes, e.g., 'x' can be stored in a simple variable of **char** type.

# String Literals

- String literals or constants are enclosed in double quotes `""`. A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.
- You can break a long line into multiple lines using string literals and separating them using white spaces.



# The #define Preprocessor


- Define keyword use to define any name or value at the start of the program.
- #define LENGTH 10

# The const Keyword

- The const keyword use to define constant value of any variable this means no one made change value of this variable at run time.
- `const int I = 10;`
- Note that it is a good programming practice to define constants in CAPITALS

# Operators

- An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators

- 
- Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators
  - Assignment Operators
  - Misc Operators

# Arithmetic Operators

- The following table shows all the arithmetic operators supported by the C language. Assume variable **A** holds 10 and variable **B** holds 20 then.

Sr.	Operator	Description	Example
1	+	Adds two operands.	$A + B = 30$
2	-	Subtracts second operand from the first	$A - B = -10$
3	*	Multiplies both operands	$A * B = 200$
4	/	Divides numerator by de-numerator.	$B / A = 2$
5	%	Modulus Operator and remainder of after an integer division	$B \% A = 0$
6	++	Increment operator increases the integer value by one.	$A++ = 11$
7	--	Decrement operator decreases the integer value by one.	$A-- = 9$

# Relational Operators

- The relational operators use to check relationship between two variables.
- The following table shows all the relational operators supported by C. Assume variable **A** holds 10 and variable **B** holds 20 then.

Sr.	Operator	Description	Example
1	==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
2	!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
3	>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
4	<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
5	>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
6	<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.



# Logical Operators

- The logical operators use to check multiple conditions in the same statement.
- Following table shows all the logical operators supported by C language. Assume variable **A** holds 1 and variable **B** holds 0, then.

Sr.	Operator	Description	Example
1	&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
2		Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A    B) is true.
3	!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

# && Operator Table

Sr.	Condition 1	Condition 2	Output
1	True	False	False
2	False	True	False
3	False	False	False
4	True	True	True

# || Operator Table

Sr.	Condition 1	Condition 2	Output
1	True	False	True
2	False	True	True
3	False	False	False
4	True	True	True

# Assignment Operators


Sr.	Operator	Description	Example
1	=	Simple assignment operator. Assigns values from right side operands to left side operand	$C = A + B$ will assign the value of $A + B$ to $C$
2	+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand	$C += A$ is equivalent to $C = C + A$
3	-=	Subtract AND assignment operator.	$C -= A$
4	*=	Multiply AND assignment operator.	$C *= A$
5	/=	Divide AND assignment operator.	$C /= A$
6	%=	Mod AND assignment operator.	$C \% = A$

# Turnery Operator

- Conditional Operator use to check condition. And also known as question mark Colon operator.
- (conditional expression ? True part : False part)

# Practice From This Unit.

1. Print Hello word.
2. Print your resume.
3. Print star pattern.
4. Single line comment.
5. Multiline comment.
6. Create and print simple integer variable.
7. Sum of 2 values.
8. Kilometer to meter convertor.
9. Simple Result.
10. Scan Value from user.

- 
11. Swap two values
  12. Swap two values without third variable.
  13. Simple interest calculator.
  14. Arithmetic Operators.
  15. Increment Operator
  16. Decrement Operator
  17. Ternary Operator.
  18. Demonstrate Constant variable
  19. Demonstrate define pre processor
  20. W.A.P to Perform size of Operator.