C Programming
Day2

Day2
Today We discuss the following topics:-
--Why not English Why C ?
--History Of C
--English Vs Programming Language
--Built in Words(Keywords)
--User defined words(Identifiers)
--Variables
--Data Types
--Constant
--Operators
--Format Specifiers
--Console I/O functions
--Simple C program
--Escape Sequences
--Precendance,Associativity and Arity
--Type Casting

Why not English why C?

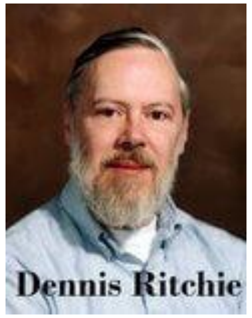
- Natural Language are ambiguous by nature.

--Bank-a financial institution or an edge of a river.

--Good-can mean useful or pleasing

- Programming language are distinct and clear
- Each word in a programming language has one & only one meaning. No two words mean the same.

History Of C

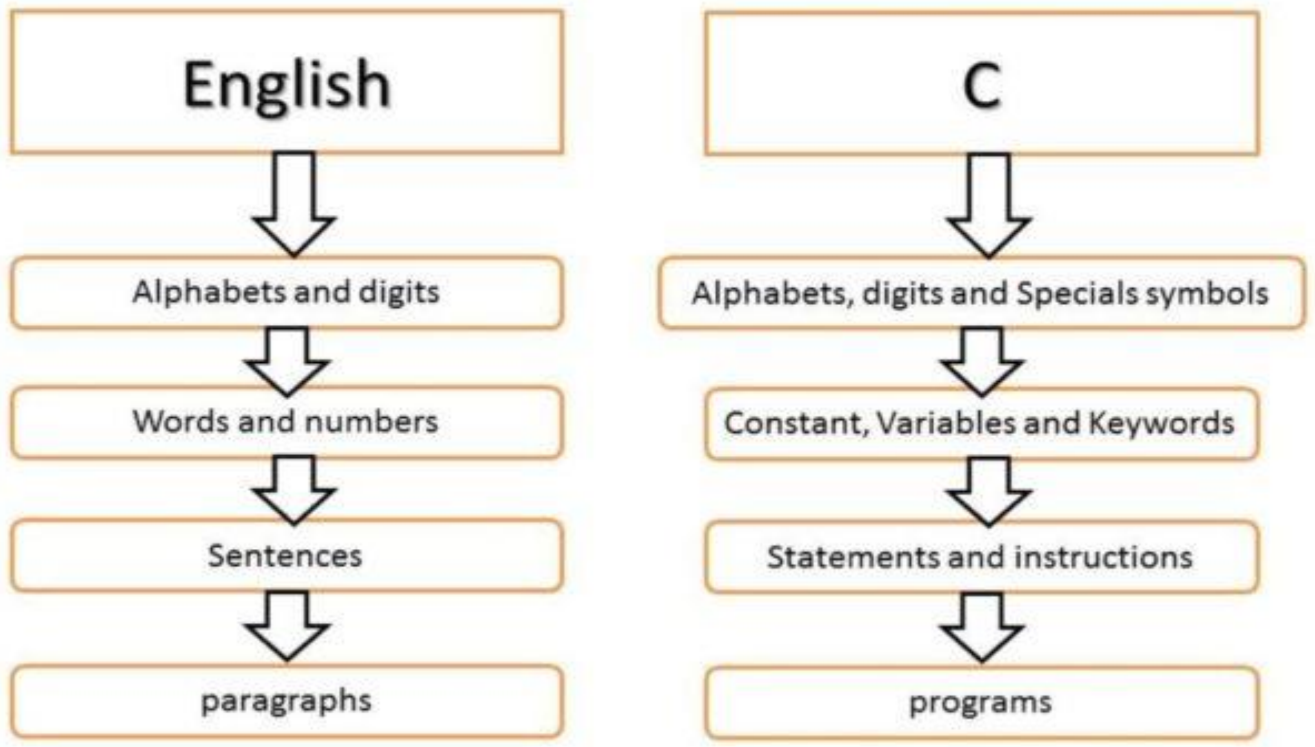


- **C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.
- **Dennis Ritchie** is known as the **founder of the c language**.
- It was developed to overcome the problems of previous languages such as B, BCPL, etc.
- Initially, C language was developed to be used in **UNIX operating system**. It inherits many features of previous languages such as B and BCPL

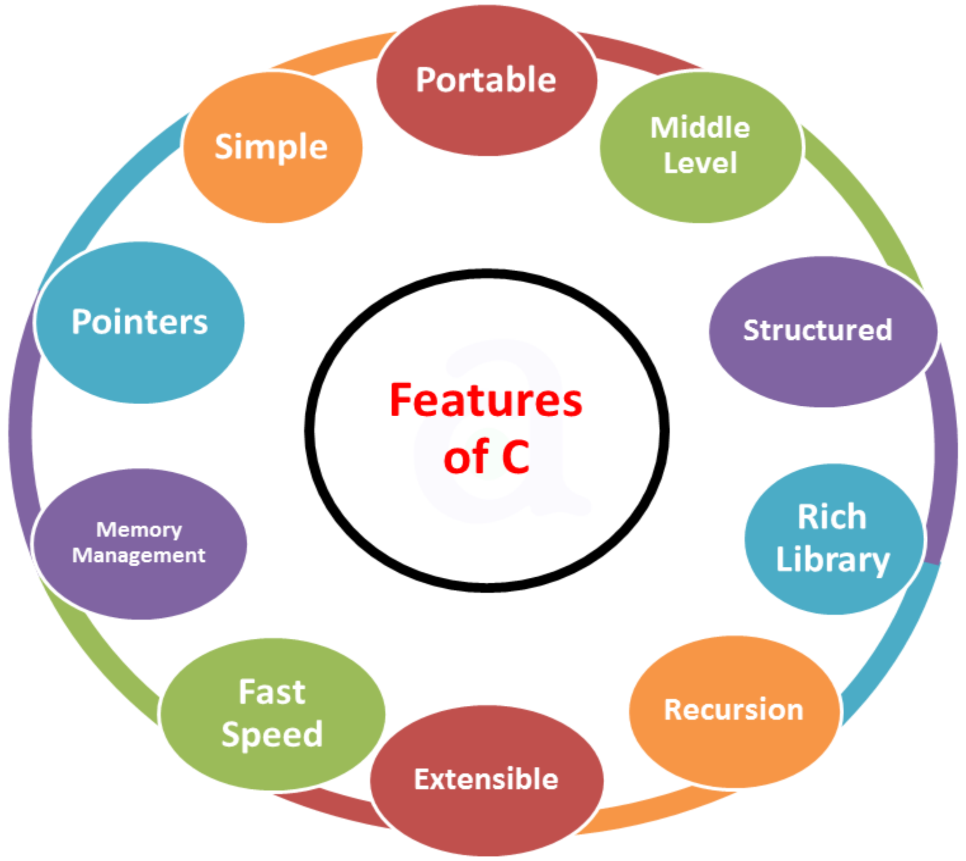
Before C Language:-

Language	Year	Developed By
Algol	1960	International Group
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
K & R C	1978	Kernighan & Dennis Ritchie
ANSI C	1989	ANSI Committee
ANSI/ISO C	1990	ISO Committee
C99	1999	Standardization Committee

English Vs C


English
Alphabets and digits
Words and numbers
Sentences
paragraphs
C
Alphabets, digits and Specials symbols
Constant, Variables and Keywords
Statements and instructions
programs

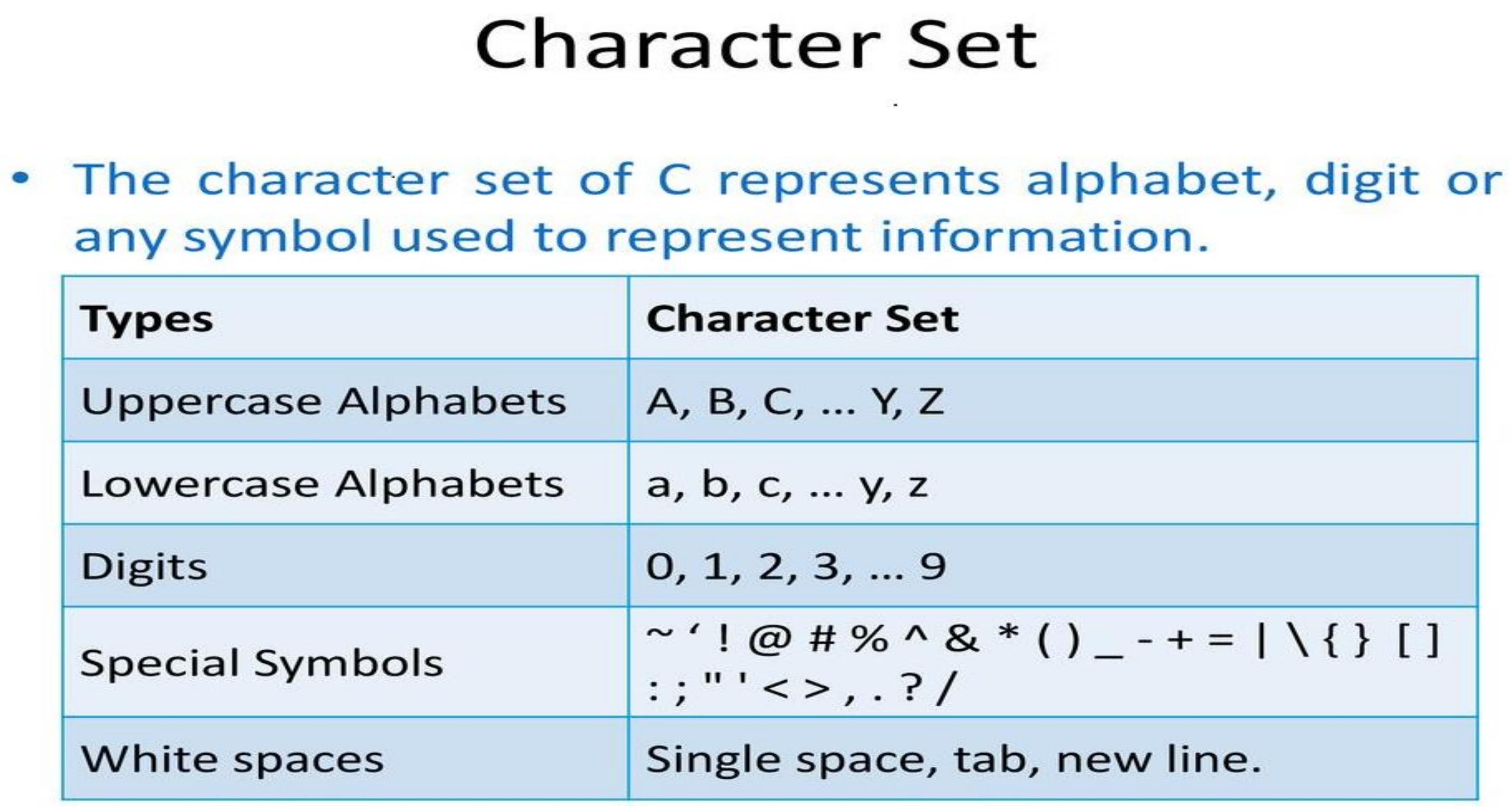
Features Of C:-



C Tokens

- Keyword
- Identifiers
- Variables
- Data Types
- Character set
- Constant
- Operators

Character Set In C:-



Character Set

- The character set of C represents alphabet, digit or any symbol used to represent information.

Types	Character Set
Uppercase Alphabets	A, B, C, ... Y, Z
Lowercase Alphabets	a, b, c, ... y, z
Digits	0, 1, 2, 3, ... 9
Special Symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /
White spaces	Single space, tab, new line.

Keywords in C

- A keyword is a **reserved word**. You cannot use it as a variable name, constant name, etc. There are only 32 reserved words (keywords) in the C language.
- A list of 32 keywords in the c language is given below:

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

C Identifiers

- C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc. An identifier can be composed of letters such as uppercase, lowercase letters, underscore, digits, but the starting letter should be either an alphabet or an underscore

Rules for constructing C identifiers:-

- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore. E.g.num1,_num1,NUM1,Num1
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Variables in C

- A **variable** is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.
- It is a way to represent memory location through symbol so that it can be easily identified.
- Let's see the syntax to declare a variable:

type variable_list;

10+20

Num1=10

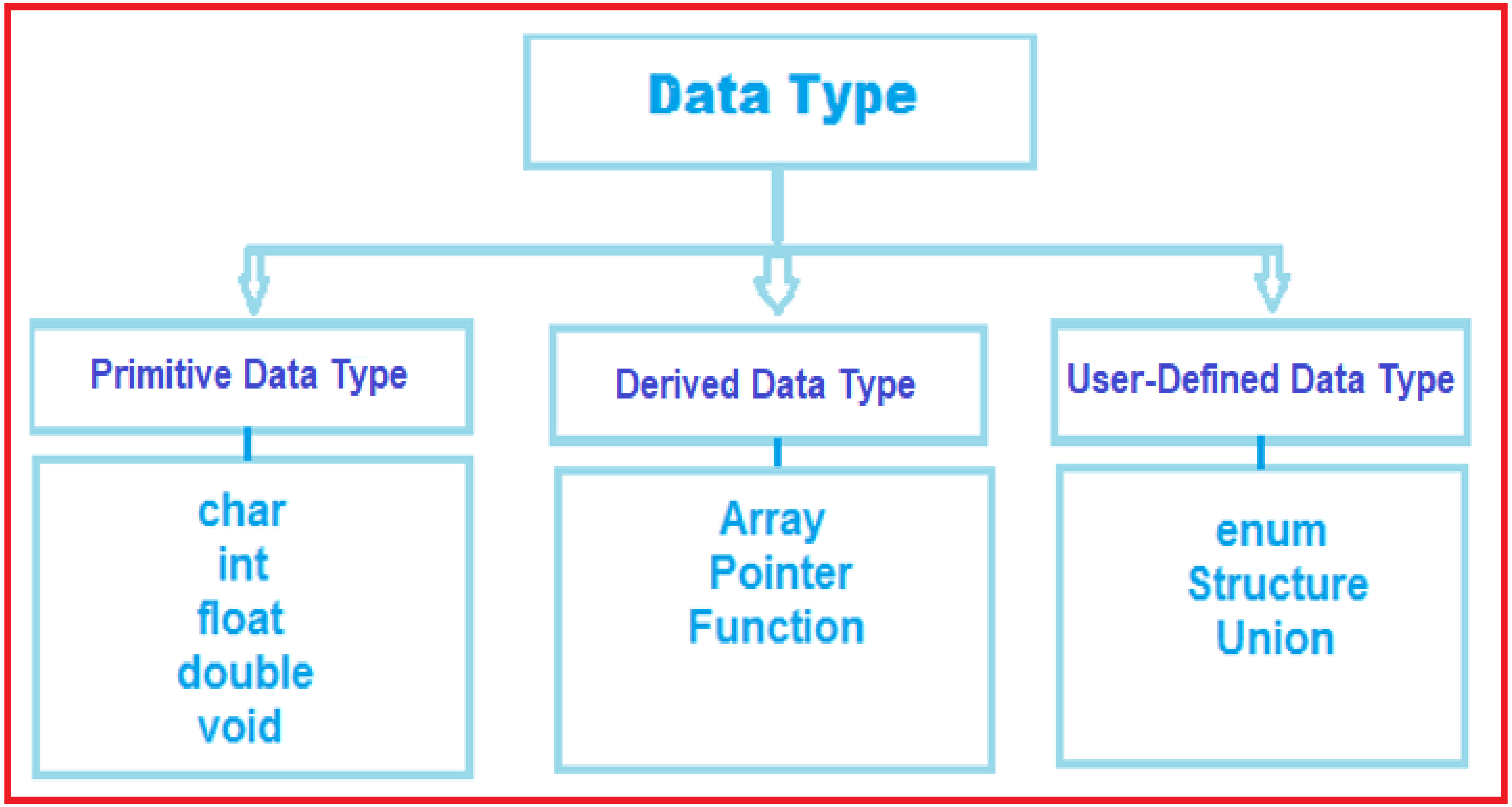
Num2=20

```
1.int a=10,b=20;//declaring 2 variable of integer type
2.float f=20.8;
3.char c='A'
```

Rules for defining variables

- Must be declared first use.
- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, float, etc.
- For local variables ,declaration statements must be placed at the beginning of the block
- For global variables, declaration statements must be placed before first function definition.

Data Types in C:-



Data Types in C

- Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. Let us briefly describe them one by one:
  Following are the examples of some very common data types used in C:
- Basic data types(Primitive)
- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- char ch1=‘a’; %c 1 byte
- **int:** As the name suggests, an int variable is used to store an integer.
- int num=10; %d 4 bytes
- **float:** It is used to store decimal numbers (numbers with floating point value) with single precision.
- float fnum=10.5; %f
- **double:** It is used to store decimal numbers (numbers with floating point value) with double precision. Double dnum=10.657; %lf

Data Types in C

- Derived data types:-

1)Array

2)Pointer

3)Function

User defined data types:-

1)Structure

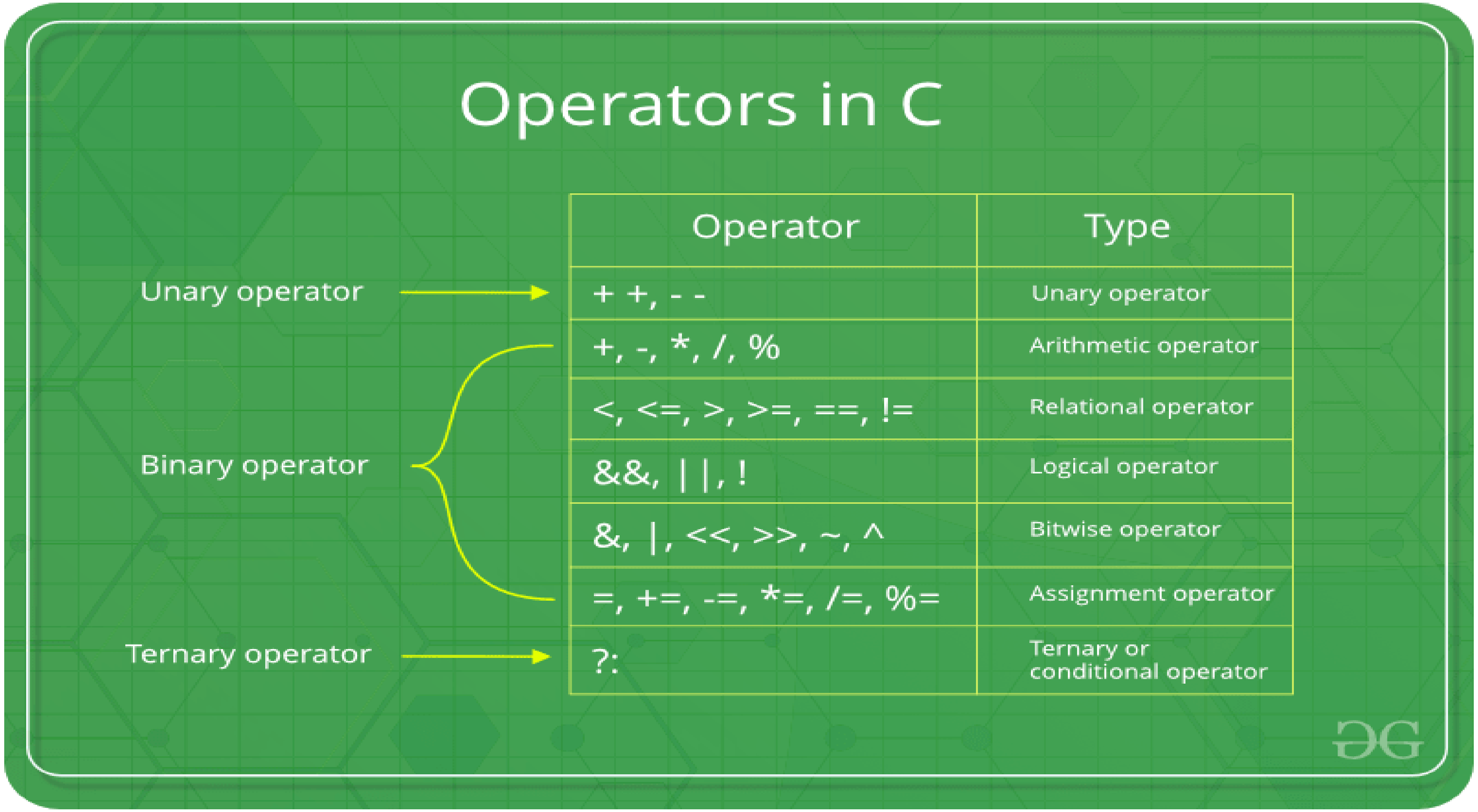
2)Union

3)Enumeration

Format Specifier

- %c—Convert data into character
- %d—Convert data into signed int.
- %f—Convert data into float(6 precisions)
- %s—Converts data into string
- %ld—Converts data into long int
- %u—converts data into unsdigned
- %x—converts data into hexadecimal format
- %o—Converts data into octal format
- %e—Converts data into exponential format

Operators in C


Operators in C
Unary operator
Binary operator
Ternary operator

Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, | |, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

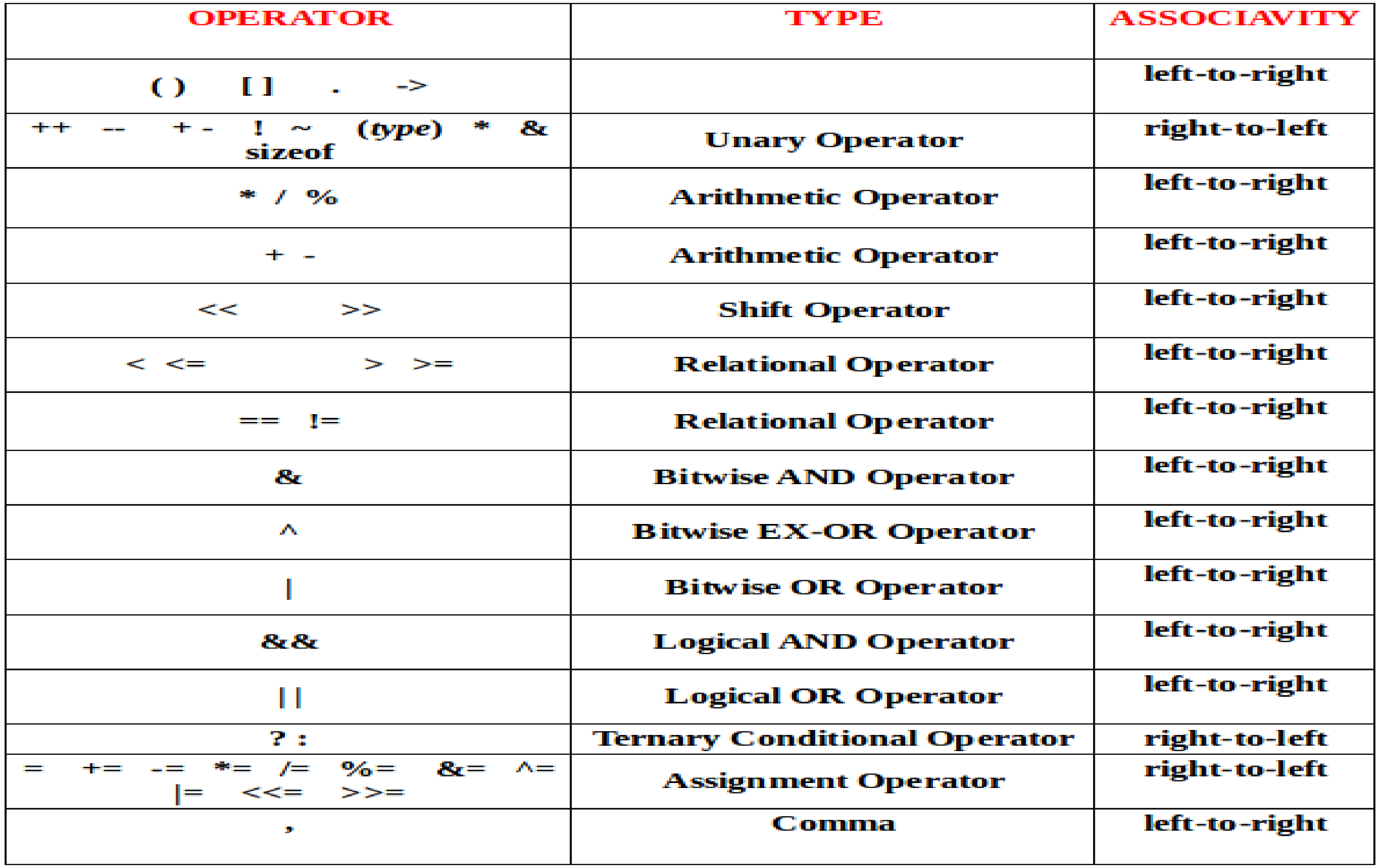
Operators in C

- **Arithmetic Operator**s (+, -, *, /, %, post-increment, pre-increment, post-decrement, pre-decrement) +
- **Relational Operators** (==, !=, >, <, >= & <=) Logical Operators (&&, || and !)
- **Bitwise Operators** (&, |, ^, ~, >> and <<)
- **Assignment Operator**s (=, +=, -=, *=, etc)
- **Other Operators** (conditional, comma, sizeof, address, redirection)

Operator Precedence and Associativity in C

- **<u>Operator precedence</u>** determines which operator is performed first in an expression with more than one operators with different precedence.
- **<u>Operators Associativity</u>** is used when two operators of same precedence appear in an expression. Associativity can be either **L**eft **t**o **R**ight or **R**ight **t**o **L**eft.
  For example: ‘*’ and ‘/’ have same precedence and their associativity is **L**eft **t**o **R**ight, so the expression “100 / 10 * 10” is treated as “(100 / 10) * 10”.

Operator Precedence and Associativity in C



OPERATOR	TYPE	ASSOCIAVITY
( ) [ ] . ->		left-to-right
++ -- + - ! ~ *(type)* * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

Comments in C

- Comments in C language are used to provide information about lines of code. It is widely used for documenting code. There are 2 types of comments in the C language.

1.Single Line Comments \\

2.Multi-Line Comments*-----------

Single Line Comments

- Single line comments are represented by double slash \\. Let's see an example of a single line comment in C.

Escape Sequence in C

- An escape sequence in C language is a sequence of characters that doesn't represent itself when used inside string literal or character.
- Character Combination consisting of a backslash(\) followed by a letter. Or combination of digits is called escape sequence.
- **The backward slash is called the escape character** because it makes Character **following it escape from its original meaning and gives special meaning to it.**

List of Escape Sequences in C

Escape Sequence	Meaning
\a	Alarm or Beep
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Tab (Horizontal)
\v	Vertical Tab
\\	Backslash
\'	Single Quote
\"	Double Quote
\?	Question Mark
\nnn	octal number
\xhh	hexadecimal number
\0	Null

Simple C Program:-

```
#include<stdio.h>
int main()
{
printf(“Hello World”);
  return 0;
}
```

Simple C Program:-

- ***Line 1: [#include <stdio.h>]*** In a C program, all lines that start with **#** are processed by a preprocessor which is a program invoked by the compiler. In a very basic term, the preprocessor takes a C program and produces another C program. The produced program has no lines starting with #, all such lines are processed by the preprocessor. In the above example, the preprocessor copies the preprocessed code of stdio.h to our file. The .h files are called header files in C. These header files generally contain declarations of functions. We need stdio.h for the function printf() used in the program.

Simple C Program:-

- ***Line 2: [***int main() ***]***
- main() is called more or less at the beginning of the program's execution, and when main() ends, the runtime system shuts down the program. main() always returns an int, as shown below:

- main() has a special feature: There's an implicit return 0; at the end.
- Thus if the flow of control simply falls off the end of main(), the value 0 is implicitly returned to the operating system. Most operating systems interpret a return value of 0 to mean "program completed successfully."

Simple C Program:-

- ***Line 3 and 6: [{ and }]*** In C language, a pair of curly brackets define scope and are mainly used in functions and control statements like if, else, loops. All functions must start and end with curly brackets.
- ***Line 4 :-***printf() is a standard library function to print something on standard output. The semicolon at the end of printf indicates line termination. In C, a semicolon is always used to indicate end of a statement
- ***Line 5 [return 0;]*** The return statement returns the value from main(). The returned value may be used by an operating system to know the termination status of your program. The value 0 typically means successful termination. .

ASCII in C

- ASCII stands for **American Standard Code for Information Interchange**.
- ASCII code is a character encoding scheme used to define the value of basic character elements for exchanging information in computers and other electronic devices.
- the ASCII code is a collection of 255 symbols in the character set, divided into two parts, the standard ASCII code, and the extended ASCII code. The standard ASCII code ranges from 0 to 127, 7 bits long, and the extended ASCII code from 128 to 255 is 8 bits long. These characters are a combination of symbol letters (uppercase and lowercase (a-z, AZ), digits (0-9), special characters (!, @, #, $, etc.), punctuation marks, and control characters. Hence, we can say, each character has its own ASCII value.

Example:-

- For example, when we enter a string as "HELLO", the computer machine does not directly store the string we entered. Instead, the system stores the strings in their equivalent ASCII value, such as '7269767679'. The ASCII value of H is 72, E is 69, L is 76, and O is 79.