Features of C Programming Language :

C Programming is widely used in Computer Technology, We can say that C is inspiration for development of other languages. We can use C for different purposes.

Below are some of the **Features of C Programming –**

1 . Structured Language

2. Block Structured

3. Procedural

4. Follows Top Down Approach

5 . Powerful :

Provides Wide verity of 'Data Types'

Provides Wide verity of 'Functions'

Provides useful Control & Loop Control Statements

6 . Bit Manipulation

C Programs can be manipulated using bits. We can perform different operations at bit level. We can manage memry representation at bit level. [Eg. We can use Structure to manage Memory at Bit Level. It provides wide verity of bit manipulation Operators. We have bitwise operators to manage Data at bit level.

7 . Modular Programming:

Modular programming is a software design technique that increases the extent to which software is composed of separate parts, called modules. C Program Consist of Different Modules that are integrated together to form complete program

8 . Efficient Use of Pointers:

Pointers has direct access to memory. C Supports efficient use of pointer .

9 . More Efficient & Fast

During execution it gives output in a correct and timely way.

Application of C Programming: Application of C Programming are listed below –

1. C language is used for creating computer applications.
2. Used in writing Embedded software.
3. Firmware for various electronics, industrial and communications products which use micro-controllers.
4. It is also used in developing verification software, test code, simulators etc. for various applications and hardware products.

5. For Creating Compiles of different Languages which can take input from other language and convert it into lower level machine dependent language.
6. C is used to implement different Operating System Operations.
7. UNIX kernel is completely developed in C Language.

Structured programming

Structured programming, sometimes known as **modular programming**.

It is a subset of procedural programming that enforces a logical structure on the program being written to make it more **efficient** and **easier to understand** and **modify**.

It is A technique for organizing and coding computer programs in which a hierarchy of modules is used, each having a **single entry** and a **single exit point**.

In Struct. Prog., control is passed downward through the structure without unconditional branches to higher levels of the structure.

Structured programming frequently employs a top-down design model, in which developers map out the overall program structure into separate subsections.

A defined function or set of similar functions is coded in a separate module or sub module, **which means that the code can be loaded in to memory more efficiently and that modules can be reused in other programs**.

Top down approach and Stepwise refinement

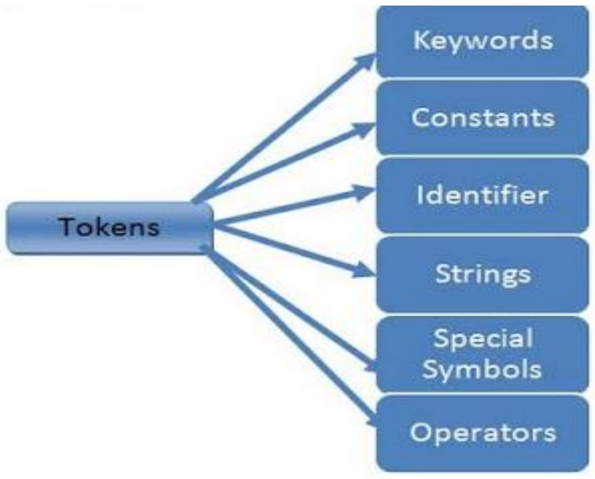
With the **top-down approach**, we start with our top-level program, then divide and further sub-divide it into many different modules. The division process is known as **stepwise refinement**.
As we design each module, we will discover what kind of sub modules we will need, and then continue to program those.
After programming our smaller sub modules, we will group them together into the larger module.

- The key to stepwise refinement is that each module should end up doing only one task.
- One advantage of the top-down approach is that we see the full picture at first.
- Therefore, the top-down approach is suitable for larger(big) problems.

C Tokens

- The smallest individual unit in a c program is known as token or a lexical unit.
- In C Programming **punctuation, individual words, characters** etc are called **tokens**.

C tokens can be classified as follows:



Token Example :

No	**Token Type**	**Example 1**	**Example 2**
1	Keyword	do	while
2	Constants	-76	89
3	Identifier or variable	Number	Sum
4	String	“HTF”	“PRIT”
5	Special Symbol	*	@
6	Operators	++	/

IDENTIFIERS

An identifier refers to the name of the object. The syntactic rules to write an identifier name in C are:

- Can have letters , digits or underscores.
- First character must be a letter or an underscore but can’t be a digit.

- No special character except underscore can be used.
- Keywords or reserved words cannot form a valid identifier name.
- Maximum number of characters that form an identifier name is compiler dependent.

C Keywords

C keywords are the words that convey a special meaning to the c compiler. The keywords cannot be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed. There are total 32 keywords defined in C

The list of C keywords 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			

declaration statement

- Every identifier (except label name) needs to be declared before it is used.
- An identifier can be declared by making use of **declaration statement**.

ROLE: To introduce the name of an identifier along with its data type (or just type) to the compiler before its use.

- The general form of declaration statement is:

→[storage_class_specifier][type_qualifier|type_modifier]type identifier [=value, identifier [, ...]];

- The terms enclosed within [] (i.e. square brackets) are optional and might not be present in a declaration statement. The type name and identifier name are mandatory parts of a declaration statement.

Examples of valid declaration in C:

- int variable; *(type int and identifier name variable present)*
- static int variable; *(Storage class specifier static, type int and identifier name variable present)*

- int a=20, b=10; (*type int, identifier name a and its initial value 20 present, another identifier name b and its initial value 10 present*)

DECLARATION STATEMENTS: 2 TYPES

SHORTHAND DECLARATION STATEMENT	**LONG HAND DECLARATION STATEMENT**
1. Declaration statements in which more than one identifier is declared is known as **shorthand declaration statement**. E.g. int a=20, b=10;	2. The corresponding **longhand declaration statements** are: e.g. int a=20; int b=10;
2. Can only be used to declare identifiers of the same type. e.g. int a=10, float b=2.3; is an invalid statement.	2. It can be used to declare identifiers of different types e.g. int a=10;float b=2.3; are valid statements.

Type-Modifiers:
It changes the base-type to get a new type**=>** range & Arithmetic props are modified.
Corresponding keywords available in C are:
1. signed
2. unsigned
3. short
4. long

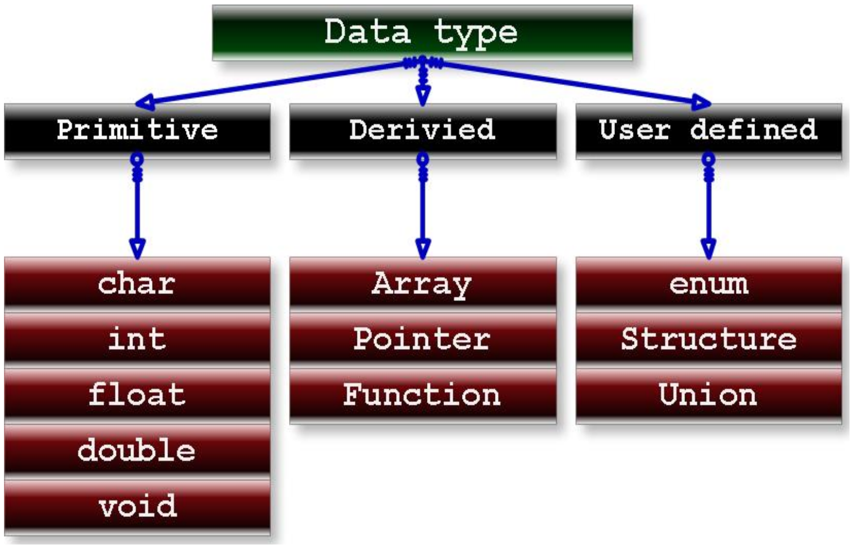
Type-Qualifiers: They are used to indicate special properties of data within an object (variable).
Two qualifiers are defined:
1.Constant (const int i=10;)
2. Volatile: The volatile keyword is intended to prevent the compiler from applying any optimizations on objects that can change in ways that cannot be determined by the compiler.

If we do not use volatile qualifier, the following problems may arise
1) Code may not work as expected when optimization is turned on.
2) Code may not work as expected when interrupts are enabled and used.

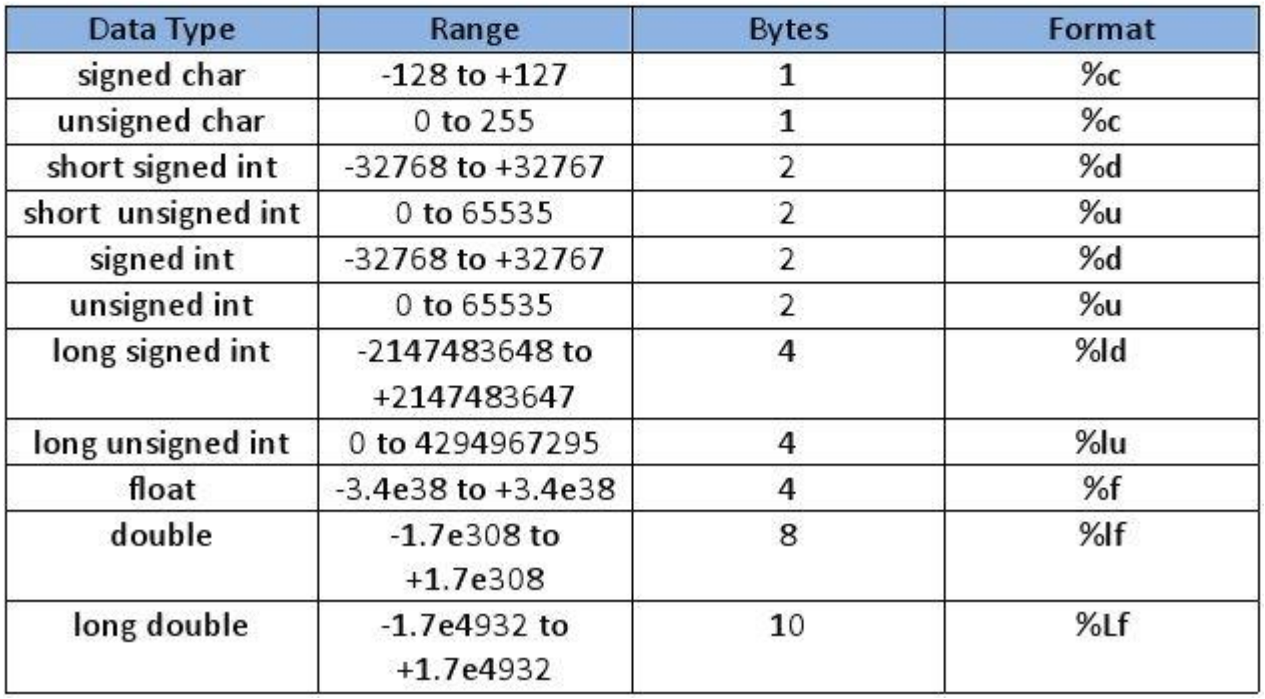
Data Object, L-value and R-value:

variables/Identifiers are used as data object which contain data in them.
LHS of '=' is called L-value (generally name of identifier/variable)
RHS of '=' is called R-value (generally value of identifier/variable)

Eg. **A= b*c+d; {b=4, c=3, d=-4}.**



Data-Types Description:



Data Type	Range	Bytes	Format
signed char	-128 to +127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf