C LANGUAGE

C language was developed by Dennis Ritchie in 1972 at Bell Telephone Laboratory in U.S.A. C was developed from a language known as BCPL (Basic Combined Programming Language later known as B language) developed at M.I.T. in the late 60's. Its popularity and tremendous growth resulted in development of different versions which were similar but incompatible to each other. Due to incompatibility in different versions of C, the ANSI (American National Standard Institute) started to define a standard for this language in 1983 which was finally approved and renamed as ANSI C in December 1989.

C language become popular due to its high level language quality i.e. machine independence and conciseness and low level language quality used for system programming. It is a general purpose structured programming language. Instructions of C language consists of algebraic terms written in simple English words like - if, else, for, while etc.

C can be used for system programming as well as application programming, hence sometime it is called middle level language. It has the simplicity of high level language as well as the power of low level language. C language provides the facilities to do programming at register level also.

Characteristics of C language: Following are the characteristics of C language:

a) **Integrity**: It refers to the accuracy of calculations in C language.
b) **Clarity:** It refers to the overall readability of the C program.
c) **Simplicity:** In C language we use simple English words for writing instructions and control statements.
d) **Efficiency:** A program written in C language executes at high speed with efficient memory utilizations.
e) **Modularity:** A large program in C language can be divided into different identifiable parts called modules, hence C language supports modularity.
f) **Stability:** ANSI C was created in 1983 and since then it has not been revised.

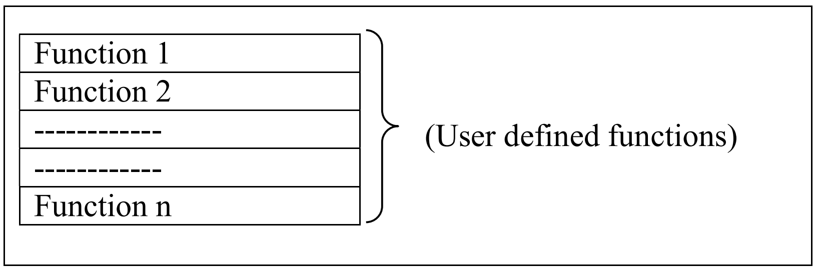
Features of C language: Following are the features of C language:

a) C language has only 32 keywords.
b) It is a robust language which contains large number of built-in functions and operators.
c) Several standard functions are also available.
d) C is highly portable language i.e. program written in C language can run on another computer with little or no modifications.
e) This is well suited for structured programming in terms of functions, modules or blocks.
f) It supports pointer to refer the computer memory which helps in accessing arrays and structures.
g) C language can be used for low level as well as high level language programming. Hence it bridges the gap of low level language and high level language.
h) C is a core language as many other programming languages (like C++, Java, Pearl etc.) are based on C.
i) C is an extensible language as it enables the user to add his own functions to the C library.

Basic structure of a C program:

C program consists of a group of blocks called functions. A function is a sub-routine that may include one or more statements depending on specific tasks. A C program has one or more sections as shown below:-

Documentations Section
Link Section
Definition Section
Global Declaration Section
main() function Section { Declaration part; Executable part; }
Sub program Section



a) Documentations Section: It consists of comment line(s) which may be used by programmer in future to know about the program. It is not a compulsory part of a program.

Structure- /*--------------*/

b) Link Section: It provides the information to the compiler to link the header file or any other file from the system library which contain library (built-in) function.

```
Example:- #include<stdio.h>
          #include<conio.h>
```

c) Definition Section: Under this section we define symbolic constant (macro).
Example:- #define MAX 20

d) Global declaration part: Variables that are used in more than one function are called global variables. Such variables are declared in global declaration section.

e) Main function section: Every C program must have one and only one main function. This section is divided into two parts- Declaration Part and Executable Part. In the declaration part we declare all the variables used in the executable part. At least one statement should be present in the executable part. These two parts must appear between the opening and closing braces ({}). The program execution begins at the opening brace and ends at the closing brace. The opening brace of the main function is logical beginning and closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semi-colon known as the termination symbol.

f) Sub program section*:* The sub program section contains the user-defined functions that are called in the main functions.

```
Structure- void main ()
               {
               function1( )
               }
             void function1( )
               {
               user defined function()
               }
```

<u>The C characters sets</u>: C has following set of characters to be used in C program. These characters in C language are grouped into the following categories:-

a) Letters – A to Z & a to z
b) Digits – 0 to 9
c) Special symbols:

Symbol	Name of symbol	Symbol	Name of symbol
,	Comma	“	quotation mark
;	semi colon	:	Colon
’	single quotation mark (left pointed)	?	question mark
(,)	Parentheses	!	exclamation mark

[,]	Bracket	|	Pipe
/	Slash	{, }	Braces
#	Hash	\	back slash
~	Tilde	.	Decimal
^	Carat	%	Percent
+	Plus	&	Ampersand
*	Asterisk	-	Minus
<	less than	_	Underscore
=	Equals to	>	Greater than
\0	Null character		

d) White Spaces: Blank spaces, horizontal tab, carriage retain, new line, backspace and form feed, audible sound(\a).

C tokens:

The basic and the smallest units of C program are called C tokens. There are 6 types of tokens in C language.

Keywords, Identifiers, Constants, Strings, Operators, Special Symbols.

1) **Keywords:** Keywords are sequence of characters (words) whose meaning has already been explained to the C compiler. Keywords can't be use as variable names. The keywords are also called reserved words. These keywords can be used only for their specified purpose. They can't be used as programmer defined identifiers. The keywords are written in lower case. There are 32 keywords which are follows:-

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

2) **Identifiers**: Identifiers refer to the names of variable, functions, arrays, structures, symbolic constants, labels etc. These are user defined name and consist of sequence of letters and digits with a letter as first character. Both uppercase and lowercase letters are allowed but both are considered of different type. Most of the identifiers are conventionally declared in lowercase. Underscore character may also be used and it can be placed at first position also. Identifiers may be of any reasonable length generally of 8 to 10 characters. Certain compilers allow longer names up to 63 characters in ASCII. Two identifiers will be considered to be different if they differ up to 31 characters.

3) ***Constant:*** A constant is a quantity that doesn't change during the execution of program. Constants are divided into two major categories:
a) Primary constants: Integers, character, float/real.
   - **Integer constant** – Integer constant is an integer number which consist of a sequence of digits. Integer constant can be used in 3 different number systems in C language which are decimal, octal and hexadecimal.
     Rules for constructing integer constant:

a) An integer constant must have at least one digit.
b) It must not have a decimal point.
c) It may be either negative or positive, default sign.
d) No commas or blanks are allowed within an integer constant.
e) The range of integer constant is -32768 to 32767.

Example:- 45, -32, 5634 etc

- **Real/floating point constant:** A floating point constant is a decimal number system that contains either a decimal point or an exponent.

  Rule for constructing real constant:

  a) A real constant must have at least one digit.
  b) It must have a decimal point.
  c) It may be either positive or negative, default sign is positive.
  d) d) No commas and blanks are allowed.

  Example:- 3.45

- **Exponential form of a real number:** In exponential representation, a real constant is represented in two parts. The part before "e" is called mantissa and the part flowing "e" is called exponent.

  Rules for constructing real constant in exponential form:

  a) The mantissa part and the exponential should be separated by letter "e".
  b) Mantissa part may have positive or negative value but default value is positive.
  c) The exponent part must have one digit which must be positive or negative. Default sign is positive.
  d) The range of real constant expressed in exponential is -3.4e38 to 3.4e38.

- **Character constant:** A character constant is a single character enclosed within single quotation mark.

  Rules for constructing a character constant:

  a) A character constant may be any character of the C character set enclosed within single inverted comma. Both the comma should point to the left.
  b) Maximum length of a character constant is one character only.
  Example: 'A', 'a', '5', '*' etc.

- **String constant:** A string constant is a sequence or group of characters enclosed in double quotation mark. The characters may be alphabets, digits, special characters or blank spaces.

  Example. " Muzaffarpur".

- **Backslash character constant/escape sequences:** These are non-printing characters which always begin with backslash and is followed by one or more special characters. Following are the backslash character constant and their meaning.

Backslash Character Constant:	Meaning:
\a	Audible sound
\b	Backspace
\n	New line
\t	Horizontal tab
\v	Vertical tab
\f	Form feed
\o	Null character
\"	Quotation mark
\?	Question mark
\'	Single inverted comma
\r	Carriage return

- **Symbolic constant:** We can use symbolic constants in a program by creating/ expanding a macro. A macro is defined with the help of #define directive in upper part (definition section) of a C program. Whenever a macro is defined in a C program, during the compilation each symbolic constant or macro is replaced by its corresponding value.

  Rules for constructing symbolic constant:

  a) Symbolic names are written in upper case letters to visually distinguish them from the normal variable which are written in lower case letters. This is only a convention not a rule.

     Example:- #define PI 3.1428
     (symbolic name) (constant value)

  b) No blank space is allowed between # and define.

  c) # must be the first character of the line.

  d) Blank space is required between #define and symbolic name and between symbolic name and the constant.

  e) #define statement must not end with semi-colon.

  f) After defining a symbolic name it must not be assigned any other value within the program by using an assignment statement.

  g) A symbolic name is not declared for its data type. Its data type depends on the type of the constant.

b) Secondary constants: Array, pointer, structure, union etc.

Data types: C language is rich in its data type. The variety of data type available in C allows the programmer to select the appropriate type according to the need of application. ANSI C supports four classes of data types.

- **Primary/fundamental data type:** The data type which represents the fundamental data i.e. characters and numbers and are already defined to the compiler is known as primary data type. Integers, characters and float are primary data types in C language. Some of these data types are further divided into different categories like signed, unsigned, short, long etc. The following chart represents the classification of basic data type and their size, symbol and range.

  Types, symbol and range of primary data type:

Type	Size (byte)	Symbol	Range
char/signed char	1	%c	-128 to 127
unsigned char	1	%c	0 to 255
short int/signed short int	1	%d	-128 to 127
unsigned short int	1	%u	0 to 255
int/signed int	2	%d	-32768 to 32767
unsigned int	2	%u	0 to 65535
long int/signed long int	4	%ld	-2,147,483,648 to 2,147,483,647
unsigned long int	4	%lu	0 to 4,294,967,295
Float	4	%f	-3.4e38 to 3.4e38
Double	8	%lf	-3.4e308 to 3.4e308
long double	10	%lf	-1.7e4982 to 1.7e4982

- ***Derived data type:*** The datatypes which are derived from primary datatypes are called derived datatype. array, pointer, structure, union etc are derived datatype.
  Ex: int a[10];
  ```
   int *ptr;
  struct mixed
      {
      int x;
      char y;
      float z;
      }v1,v2;
  ```

- ***User defined datatype:*** There are two ways to create user defined datatype-
  a) **<u>Using typedef statement</u>** – C supports a feature known as type definition that allows user to define a new name to represent an existing primary or derived datatype
  Syntax:- typedef datatype name;
  Example:- 1) typedef int integer;
  2) typedef unsigned long int uli;

  b) **<u>Using enum statement</u> -** In addition to typedef statement C allows another user defined datatype enumerated datatype which can be created using enum keyword
  Syntax:- enum datatype_name { value1, value 2, value 3,........... value n};
  Example:- 1) enum days { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
  enum days holiday = Sunday;
  2) enum months { January, ..., December};
  enum months summer = June;

  Using enum statement we create a new datatype which is further used to declare variables that can have one of the values enclosed within the braces (known as enumeration constant). Enumerated datatype are not used frequently in a program because there is no format for input – output of enumerated datatype. A variable of enumerated datatype can only be assigned with an enumeration constant and compared within a decision making statement. Hence, the use of enumerated datatype is avoided as far as possible in program.

- ***Empty data set:*** void, null, \0 are three members of empty data set.