INTRODUCTION TO ‘C’ LANGUAGE:

We know that computers can understand machine language written in 0’s and 1’s. So the computer has to be instructed or programmed using O’s and 1’s.

- The instructions given to the compiler in terms of 0’s and 1’s called Machine language.
- But this machine level language is very difficult to understand.
- To overcome these disadvantages some high level languages are introduced.

Ex: c, c++.

Features of ‘C’ language:

- ’C’ is structured programming language or procedural language.
- It is called as high level language or middle level language.
- We can develop efficient programs.
- It has less number of keywords.

- It can be executed on many different between platforms.
- It has wide variety of operators using which a program can be written easily to solve a given problem.

Background:

'C' language was introduced by Dennis Ritchie in the year of 1972. It is the combination of B&BCPL (Basic combined programming language).

'C' Tokens:

- A token is a smallest or basic unit of a 'C' program.
- One or more characters are grouped to form meaningful words. These meaningful words are called tokens.
- Tokens in 'C' language are broadly classified as
  a) Keywords
  b) Identifiers
  c) Constants
  d) Operators
  e) Special symbols

KEYWORDS:

- The tokens which have predefined meaning in ‘C’ language are called keywords.
- These keywords are reversed for specific purposes in C language. They are also called as reversed words.
- Example for keyword: if, int , float, double, long, switch, while, continue, break etc…
- The above keywords should not be used as variable name, function names etc…
- User can’t change the meaning of keywords.
- All keywords are lower case letters

IDENTIFIERS:

- Identifiers are the names given to various program elements such as variables, constants, function name, array elements etc..
- It is a word consisting of sequence of one or more letters or digit along with an underscore “_”.
- The rules to be followed to frame an identifier are:
  1. The first character in the identifier should be a letter or an underscore and can be followed by any number of letters or digits or underscore.

2. No extra symbols are allowed other than letters, digits and an underscore (“_”).
3. The length of an identifier can be up to maximum of 31 characters.
4. Reversed words or keywords cannot be used as identifiers.
5. Examples for valid identifiers: a, b, SUM1, _amount, _sub2 etc….
6. Example for invalid identifiers: 1ab, a#b, if, for etc…

CONSTANTS:

- A constant is a data item which will not change during the execution of the program.
- Constant in ‘c’ language is classified as,
  1. Numeric constant
  2. Character constant
  3. String constant
- Numeric constant:
  a) Integer constant
  b) Floating point constant
  a) Integer constant:

- An integer constant is a whole number without any fraction part.
- No extra characters are allowed other than '+' and '-'ve sign (only to precede the number).
- An integer constant can be classified into three types shown below

I. Decimal
II. Octal
III. Hexadecimal

I. Decimal number:

- A decimal integer constant can be any combination of digits from '0' to '9'.
- Decimal number should not precede by '0', comma not allowed and space not allowed.
- Examples:

  Valid : 100, -67, +989, +1000….

  Invalid : 10,000, 0.5, 10 20, 0259 etc…

II. Octal number:

- An octal integer constant can be any combination of digits from '0' to '7' with a prefix '0' (zero) and '+' & '-' ve sign are allowed as preceding characters.
- Examples:

Valid : 010, 0777, -065 etc…

Invalid : 10,000, 0.5, 085, 777 etc……

III. Hexadecimal number:

- A hexadecimal integer constant can be any combination of digits from ‘0’ to ‘9’ along with the letters ‘A’ to ‘F’ or ‘a’ to ‘f’.
- This constant has to be preceded by 0X or 0x.
- Examples:

  Valid : 0x8A, 0Xab, 0x123 etc….

b) Floating point constants:

- Floating point constants are the decimal numbers with function part such as 10.5, you can use ‘+’ and ‘-‘ve symbols.
- These floating point constants are can be represented in two ways

  I. Fractional form

  II. Scientific notation or Exponent form

  I. Fractional form:

  - A floating point numbers represented using fractional form has an integer part followed by a dot and a fractional part.

- We can omit the digits before the decimal point or after the decimal point.
- Examples: 0.5, -0.99, -0.6, -9.0, +9.0 etc…

II. Scientific form or Exponent form:

- The floating point number represented using scientific notation (also called exponential notation).
- Examples:

Mantissa	e/E	Exponent	Result
9.81	E	3	$9.81*10^3$
9.81	e	3	$9.81*10^3$

Where, the mantissa can be an integer or a floating point number represented using fractional notation.

- Character constant:
  - A symbol enclosed within a pair of single quotes is called a character constant.
  - Each character is associated with a unique value called an ASCII (American standard code for information interchange) value.
  - Examples:

Valid : ‘9’, ‘+’, ‘$’, ‘\n’, ‘\t’ etc….

Invalid : “9”, ‘99’ etc...

- String constant:
  - A sequence of characters enclosed within a pair of double quotes is called string.
  - The string always ends with null (‘\0’) character.
  - Examples:

    Valid : “99”, “ABC”, “ “, “PRASANNA” etc….

    Invalid : ‘9’ etc….

BASIC/SIMPLE/PRIMITIVE Data types:

- The data type defines the type of data stored in a memory location.
- The data type determines how much memory should be allocated for a variable.
- Primitive data types are,

1. int
2. char
3. float
4. double
5. void

1. int

- An int is a keyword which is used to define integers in ‘c’ language.
- Using this keyword, the compiler determines the size of the data to be manipulated and reserve space in memory to store the data.
- The size of int is machine dependent.
- Size and range of integer varies from machine to machine.

Size of data	Range of unsigned int	Range of unsigned int
2 bytes	0 to 2^{16}-1 (0 to 65535)	-2^{16-1} to $+2^{16-1}$-1 (-32768 to +3276
4 bytes	0 to 2^{32}-1	-2^{32-1} to 2^{32-1}-1

2. float

- A float is a keyword which is used to define floating point numbers in ‘C’ language
- The compiler can determine the size of the data to be manipulated and reserve space in the memory to store the data.
- The size and range of float

Machine	Size of float	Range of float

16 bit	4 bytes	3.4 E-38 to 3.4 E+38
32 bit	8 bytes	1.7E-308 to 1.7E+308

- Note:- In floating point numbers maximum 6 digits are allowed after the decimal point.

3. double

- A double is a keyword which is used to define long floating point numbers.

Machine	Size of double	Range of double
16 bit	8 bytes	1.7E-308 to 1.7E+308
32 bit	16 bytes	3.4E-4932 to 1.1E+4932

4. char

- A char is a keyword which is used to define single character or a sequence of characters called string in ‘c’ language
- The size and range of characters

Machine	Size	Unsigned char range	Signed char range
16 /2 bits	1 byte	0 to 2^8-1 ie 0 to 255	-2^7 to $+2^7$-1 ie -128 to +127

5. void
   - It is an empty data type. Since no value is associated with this data type, it does not occupy any space in memory.
   - The size is 0 and it has no values for range.

VARIABLES:

- A variable is a name given to a memory location where the data can be stored.
- Using the variable name, the data can be stored in a memory location and can be accessed or manipulated very easily.
- Rules for defining a variable
  1. The first character in the variable should be a letter or an underscore and can be followed by any number of letters or digits or underscore.
  2. No extra symbols are allowed other than letters, digits and an underscore (“_”).
  3. The length of the variable can be up to maximum of 31 characters.

4. Reversed words or keywords cannot be used as variable name.

- Examples:

  Valid **: principle_**amount, a , ___a___b, a123, _ab etc….

  Invalid : 1abc, for , 3$2# etc….

➢ <u>Defining a variable</u>

- Defining a variable is a method of informing the compiler to reserve the memory space for the data based on the type of variable.
- Syntax:-

  type $v_1, v_2, \ldots v_n$;

  ➢ Type is any basic data type.
  ➢ $V_1, v_2, v_3, \ldots v_n$ are the variable names.
  ➢ Each variable should be separated by a comma.
  ➢ Eg:- int a, b, c;

  Where int is type and a,b,c are the variables.

- We can define more than one variables of same type in single line.

  Eg:- int a, b, c ;

float b,c,d;

➢ Variable initialization:-

- When a variables are defined they are not initialized to any value, they contain garbage value
- The method of giving the initial values for the variables before they are processed is called initialization of variables .
- Initializing one variable:-

  type variable name =value;

  Example: int a=10;

- Initializing more than one variable,
  int count=0, SUM=10, a=10, b=20;

STRUCTURE OF 'C' PROGRAM:

PREPROCESSOR DIRECTIVES
GLOBAL DECLARATION
main() {

```
    Local declaration;
    Statement1……..;
    Statement2……..;
                    ……;
                    ……;
                    ….n;
}
```

- PREPROCESSOR DIRECTIVES:

- Always written in the first line of program which prepare source program for compilation.
- These directives are started with a symbol #, normally all program used a directives called include which include specified file contents at the beginning of program.
- Example: #include<stdio.h>

- GLOBAL VARIABLES (DECLARATION):

    - These are the variables which are declared after preprocessor directives.
    - These variables available through out the program.

➢ main():

- Every ‘c’ program should have main(). This is the first function to be executed always.
- { - Left braces indicate the starting of main function.
- } - Right braces end of the main function.
- Within the above braces statements are enclosed it is called body of the main function.

➢ CREATING AND EXECUTING THE PROGRAM:

- Any programs are created and executed in an IDE(Integrated development Environment).
- For “c” program we can use Turbo c++ IDE to create and execute program.
- Steps in turbo c++ IDE to create and execute.
  Step 1: Create a new file in the Editor and save that file with .c extension.
  Step 2: Type program in the editor and save your program.
  Step3: Compile program by pressing Alt+F9

Step4: Execute program (run program) by pressing Ctrl+F9.

INPUT AND OUTPUT FUNCTIONS

Input and output functions in “c” language are mainly classified in to two major categories,

1) Formatted I/P and O/P function.
2) Unformatted I/P and O/p function.

1) FORMATTED I/P AND O/P FUNCTION:

- FORMATTED INPUT FUNCTION:

➢ Formatted input function is scanf().
➢ Formatted input functions are used to read integers, floating point numbers and characters.
➢ The general syntax of the scanf() function is,

Scanf (“format specifiers”, &v_1, &v_2, …..&v_n);

➢ The scanf function has two parameters.

1. Format specifiers: It consist of one or more format specifier, begins with % symbol.

Ex: %d – Integers
%f – floating point numbers
%c – characters

%s – strings

%lf – double

%ld – long integers

2. &v_1, &v_2, &v_3,……&v_n: these are the list of variables which stores the value given by the user.

   Ex: scanf(“%d”,&a);

   where %d is format specifiers and ‘a’ is variable to store input.

- Read floating point numbers:

  scanf(“%f”,&a);

- Read a character:

  scanf(“%c”,&a);

- Read a string:

  scanf(“%s”,&str);

- Read multiple values in a single scanf function:

  scanf(“%d %f %c”, &a, &b, &c);

- FORMATTED OUTPUT FUNCTION:

- printf() is a formatted output function which is used to print any type of values such as integer, floating point, double, character and string etc…
- The function has two arguments, the general syntax

  printf("string", v1, v2,…..vn);

  where string – any message or format specifiers.

  v1, v2,….vn – variables or expression.
- Example:

  printf("WELCOME");

  printf("I am a student of %d standard", a);

  In the above printf statement. In the output screen in the place of %d the value associated with a will be print.

  printf("%d %f %c", a, b, c);

2) <u>UNFORMATTED INPUT AND OUTPUT FUNCTION:</u>

- <u>UNFORMATTED INPUT FUNCTION:</u>
- These are four input functions
  1. getchar()
  2. getch()
  3. getche()
  4. gets()

1. getchar():
   - It is an unformatted input function which is used to read a character.
   - Example:

     Char ch;

     Ch=getchar();

     After executing above statement ‘a’ is associated with ch.
2. getch():
   - Entered character are not visible.
3. getche():
   - Entered character are visible.
4. gets():
   - This is an unformatted input function which is used to read a string(sequence of character).
   - Example:

     char s[10];

     gets(s);

- <u>UNFORMATTED OUTPUT FUNCTIONS:</u>

1. putchar()
2. puts()

1. putchar():-

- It is an unformatted output function which is used to print a character.
- Example:

```
char ch[10];
ch=getchar();
putchar(ch);
```

2. puts():-
   - In this unformatted output function we can print a string (sequence of characters).
   - Examples:

```
char ch[10];
gets(ch);
put(ch);
```

- LIMITATIONS OF UNFORMATTED I/P AND O/P FUNCTIONS:

➢ We can read and write only characters.

- ADVANTAGES:

➢ It is very easy to use compare to formatted.

- DIFFERENCE BETWEEN SCANF() AND GETS():

- scanf():- It will not accept the characters after a space.
- Example: if type PRASANNA COLLEGE
  It accepts only PRASANNA and it leave COLLEGE.
- gets():- It accepts all the characters until we press enter key.