Introduction to C Language

Introduction to C

- C is a general-purpose programming language developed by ***Dennis Ritchie*** at A T & T Bell laboratories in 1972.
- C is closely related to UNIX system and many of the important ideas are taken from "BCPL" (Basic Combined Programming Language) developed by Martin Richards and "B" language.
- C provides variety of data types, derived data types like pointers, arrays, structures and unions.
- It also provides control flow constructions like conditional and looping constructs etc..,

Why C?

- Generally, computer understands only binary data i.e., 0's and 1's.
- Writing and understanding programs in binary was very difficult and time consuming.
- To overcome this middle level language was introduced; the user feels comfortable in writing the program.
- Many middle levels languages are available like C, C++ etc.
- Even though the program is written in middle level language, the code has to be converted to machine language so that computer can execute it for which we make use of *compiler*.

Advantages/Features of C Language

There are many features which attracted programmers for the usage of C. They are

1. **Simple and Portable:** it gives programmer access to all functions of the machine and can work on different kinds of computers.
2. **Powerful and Flexible:** most of the functionalities like UNIX is written in C. The compiler and interpreter are also written in C language.
3. **Modularity:** C is a structured programming language which mainly deals with the procedures. It is also termed as procedure-oriented language.
4. **Efficient:** C is more efficient which increases the speed of execution and management of memory compared to low level languages.
5. **Programmer oriented:** It has many data types and flexible control structure which gives access to hardware.
6. **Other features include**
    - Machine independent
    - No need to worry about machine details.

Characteristics of C:

- C is a middle level language.
- C is a structured language.
- C is a programmer's language.
- C is a *case sensitive* language.

Versions of C:

The main is ANSI C (accepted by American National Standards Institute) under which we have

- ✓ Lattice C
- ✓ Microsoft C
- ✓ Quick C
- ✓ Turbo C
- ✓ Borland C
- ✓ Small C
- ✓ Tiny C
- ✓ Online Compilers

Developing Programs in C:

1. **Creating the Program:**

   The source program can be written or modified using editor and can be saved with the extension of .C

2. **Compiling the program.**

   Where the program is checked for the Syntax and Semantics of the language and reports the errors to the user. These errors have to be corrected by the users in the source program.

3. **Executing the Program.**

   Once the error free code is generated after the compilation process, the program can be executed with valid inputs by linking to the required header files.

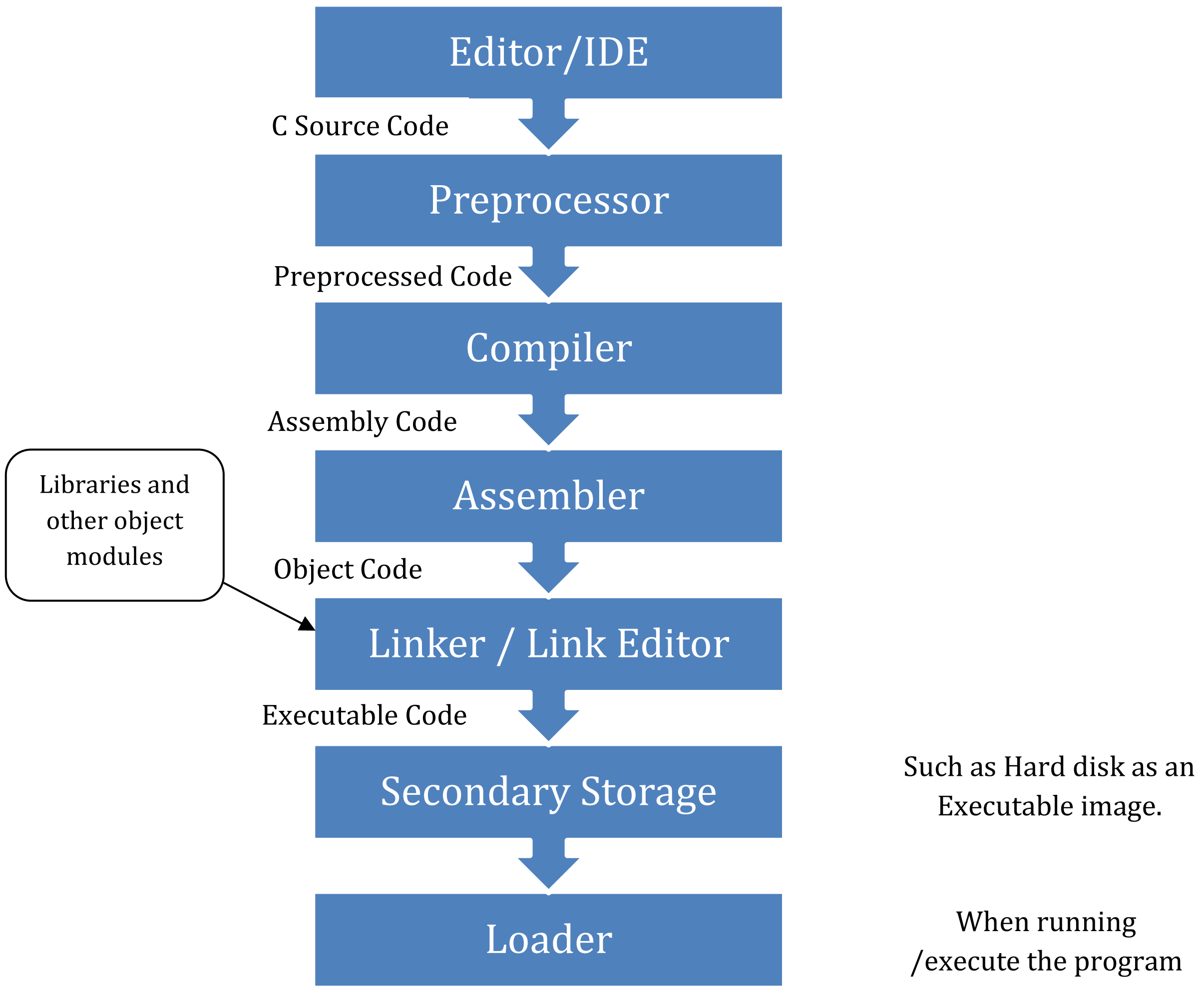
   For the successful execution of the C Program some of the software components are required they are:

   - Operating System.
   - Text Editors.
   - Compilers.
   - Assembler.
   - Loader.
   - Linker.

   On the successful execution of the C Program the following files are created:

1.	**Pgm.C:**	Contains source code of the program saved in .C extension.
2.	**Pgm.txt:**	Contains text format of the source code stored in .txt a notepad file.
3.	**Pgm.obj:**	Contains the object code of source program in terms of 0's & 1's.
4.	**Pgm.exe:**	An Executable file generated after successful compilation.
5.	**Pgm.bak:**	An backup file for the Pgm.C

Typical steps for Entering, Compiling and Executing C programs



Error and Its Type

Error is also known as bug in the world of programming, it may occur unwillingly which may prevent the program to compile and run correctly as per the expectation of the programmer. Programmer may come across with the following types of errors.,

- **Syntax Error:** errors due to the fact that the syntax of the language is not respected.

  Example: int a=20 // Error: Statement missing (;)

- **Semantic Error:** errors due to the improper use of program statements.

  Example 1.int i; // Error: Static semantic error @ compile time.
  i++; // Error: Variable "i" is not initialized.

  Example 2.int a[10]; // Error: Dynamic semantic error @ run time.
  a[10]=40; // Error: Invalid Initialization of 11 position with value 40.

- **Logical Error:** errors due to the fact that the specification is not respected.

  Example: // Program for addition of two numbers.

  ```
  int  a=20, b=10, c;
  c=a-b;
  ```

  // Error: in logic subtraction is done instead of addition.

At the time of error detection, it is classified in to two types:

- **Compile Time Errors:** Syntax and Static Semantic errors indicated by the compiler.
- **Runtime Errors:** Dynamic semantic errors and logical errors that cannot be detected by the compiler.

Note:

- **Debugging:** The process of identifying the errors and correcting it.
- **Source Code:** The text of a program that a user can read, modify and given as a input for the C compiler.
- **Object Code:** Translated code generated by the compiler by taking source code as the input and computer can executes it directly.
- **Compile Time:** The time during which your program is being compiled.
- **Run Time:** The time during which your program is executing.
- **Short Cut keys (Turbo C++ IDE):**
  - Menu: F10.
  - Saving: Alt + F2.
  - Opening the Existed program: F3.
  - Compilation: F9.
  - Execution: Ctrl + F9.
  - To see the previously Executed screen: Alt + F5.
  - Single Tab View: F5.
  - To Maximize the Screen: Alt + Enter.

Concepts of C Programming

Structure of C Program

```
Comments/Documentation Section
-----------------------------------
Preprocessor Directives
-----------------------------------
Global Declaration Section
-----------------------------------
void main()  [Program Header]
{
Local Declaration part
        Execution part
         Statement 1
           ------------
           ------------
         Statement n
}
-----------------------------------
User Defined Functions
```

Comments/Documentation Section

- Describes the program, which helps in understanding the program easily.
- The comments are not mandatory.
- The comment are may be of:
  - ✓ Single line Comment:
    - Description is written in single line with the delimiter //.
    - Example: //Program to multiply two numbers.
  - ✓ Multi line Comment:
    - It is written in more than one line.
    - Begins with /* and ends with */. The symbols /* and */ are called comment line delimiters
    - Example:
      /*Program to find the roots of the Quadratic equation on checking for value of a not equal to 0 and condition for roots are real and equal, real and distinct & Imaginary*/

Note: Comments are ignored by C compiler, i.e., everything within comment delimiters

Preprocessor Directives

- Preprocessor Directives begins with a # symbol.
- It preprocesses the include files, conditional compilation instructions and macros before it encounters the main () function.
- A function is a building block of a program where it performs a particular task, on execution it should return some type of value to where it is called.
- Some examples of preprocessor statements are:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#define PI 3.142         Etc…
```

Global Declaration Section

- There will be some variable which has to be used throughout / anywhere in program i.e.., the scope of the variable is in more than one function can be declared in global Declaration Section.

The Program Header

- Every program must contain a main() function. The execution always starts from main function.
- Every C program must have only one main function.
- void is the return type mentioned with main( ) , which represents main( ) returning nothing.

Body of the program

- Series of statements that performs specific task will be enclosed within braces { and }
- It is present immediately after program header.

  It consists of two parts

  1. **Local Declaration part**: Describes variables used in the program

     Ex:
     ```
     int sum = 0;
     int a , b:
     float b;
     ```

  2. **Executable part**: Contains input statements, processing statements (logic) & output statements to carry out some particular task.

  Note: All statements should end with semicolon(;) a statement terminator.

User Define Function / Subroutine Section:

- All user defined functions that are called in main function should be defined.

C Programming Concepts

- A program is a group of instructions given by the programmer to perform a specific task.
- To achieve programming there are many basic components of programming that has to be learnt.

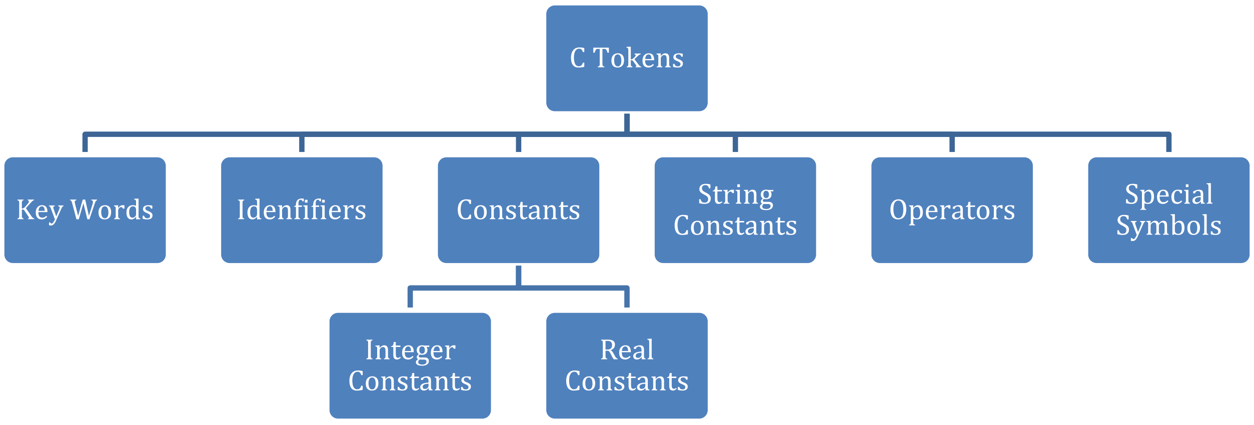
Character set of C language

The C language includes the characters.

- Alphabets(lowercase a-z, Uppercase A-Z)
- Digits (0-9)
- Special Symbols(~ ‘ ! @ # % & * () - + / $ = \ { } [ ] : ; “ “ ? etc)

C- Tokens

- Tokens are the smallest individual units of C program.
- A token is collection of characters.
- Some of the C- Tokens available in C are:



Keywords

- Words which have predefined / fixed meanings are called keywords.
- These are basic building blocks of C program statements.
- The meaning of keywords will be known to computer no new name or meaning can be defined for it.
- There are totally 32 keywords supported in C they are:

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

Important points to be remembered about keywords

- *Keywords should be written in lowercase letters.*
- *Keywords meaning cannot be changed by the users.*
- *It should not be used as variables, functions names and array names.*

Identifiers (Variables)

- Identifiers are the name given to the program element.
- The program element may be identified as variables, functions and arrays etc..,
- These are user defined names and do not have fixed meaning.
- It is name given to computer memory location.

Variables – Variables are the identifiers, the named memory location whose value changes on program execution.

Rules for identifiers/variables

- First character must be an alphabet or an underscore(_)
- First character is followed by any number of letters or digits.
- Only first 31 characters are significant
- Keywords cannot be used as identifier / variable.
- Must not contain blank space/white space
- Must contain only alphabets, digits and one special symbol underscore (_)

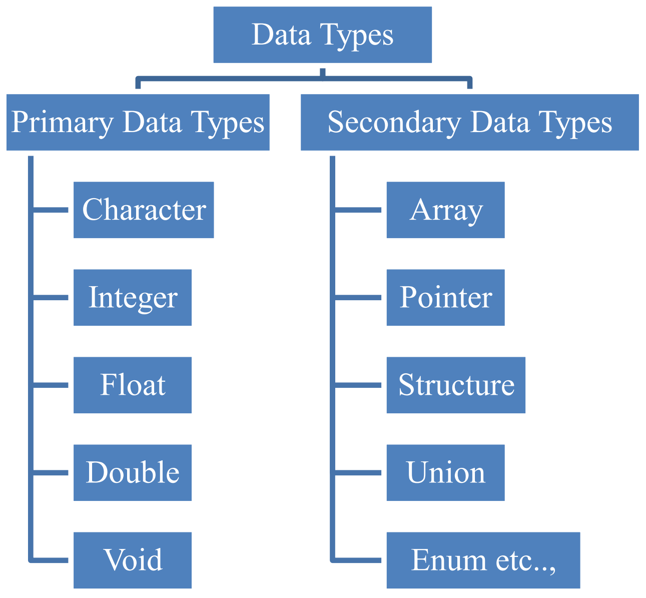
Ex:-

india	Valid
india06	Valid
_india	Valid
india_06	Valid
india_06_king	Valid
_ _india	not valid as it contains successive underscore
06india	not valid as it starts from digits
Int	not valid since int is a keyword
india 06	not valid since there is space between india and 06
india@06	not valid since @ is not allowed

Data types

- The data type defines the type of data stored in memory location or the type of data the variable can hold.

Data Types classification:



- Primary data types – are the fundamental data types which are readily available in C.
- Secondary data types – are the derived types with the help of primary data types.

Character data type (char):

- Capable of holding one character from the local character set.
- The size of the character variable is 1 byte.
- The range is from -128 to +127.
- Each character stored in the memory is associated with a unique value termed as ASCII (American Standard Code for Information Interchange).
- It is used to represent character and strings.

Integer data type (int):

- It stores an integer value or integer constant.
- Typically reflects the natural size of integers on host machine.
- General size is 2 bytes.
- The range is -32768 to +32767 (To calculate the size 1 byte =8bits so 2bytes will be 16 bits then range will be -215 to +215 -1).
- int can have additional data type which is long int whose size will be of 4 bytes and the size ranging from -231 to +231 -1.

Floating data type (float):

- Single precision floating point.
- Holds a real constant.
- The size is 4 bytes.
- The range is -3.4e38 to +3.4e38.
- Float can hold the real constant accuracy up to 6 digits after the decimal point. That is 23.456789 is valid where as 45.678953231 is invalid.

Double data type (double):

- Double precision point.
- Holds real constant.
- The size is 8 bytes.
- The range is from -1.7e308 to +1.7 e308.
- Double can hold real constant accuracy up to 16 digits after the decimal point.

Void data type (void):

- It is an empty data type.
- No memory is allocated.Mainly used when there are no return values on function execution.

Table Contains All Data Types Defined by the C Standard

Sl. No	Data Type	Sizes in		Range
		Bits	Bytes	
1	char	8	1	-128 to 127
2	unsigned char	8	1	0 to 255
3	signed char	8	1	-128 to 127
4	int	16	2	-32,768 to 32,767
5	unsigned int	16	2	0 to 65,535
6	signed int	16	2	-32,768 to 32,767
7	short int	16	2	-32,768 to 32,767
8	unsigned short int	16	2	0 to 65,535
9	signed short int	16	2	-32,768 to 32,767
10	long int	32	4	-2,147,483,648 to 2,147,483,647
11	long longint	64	8	-(2^{63} -1) to (2^{63} -1) (Added by C99)
12	signed long int	32	4	-2,147,483,648 to 2,147,483,647
13	unsigned long int	32	4	0 to 4,294,967,295
14	unsigned long longint	64	8	2^{64} – 1 (Added by C99)
15	float	32	4	-3.4E38 to +3.4E38 with six digits of precision
16	double	64	8	1.7e308 to +1.7 e308 with ten digits of precision

Format Specifiers

- Format specifiers are the character string with % sign followed with a character. It specifies the type of data that is being processed. It is also called conversion specifier. When data is being output or input it must be specified with identifier (variable) and their format specifier.
- There are many format specifiers defined in C. Take a look at the following list:

Symbols	Meaning
%i or %d	integer number
%f	float point number
%c	Character
%o	octal number
%x	hexadecimal integer(Lower case letter x)
%X	hexadecimal integer(Upper case letter X)
%e	floating point value with exponent(Lower case letter e)
%E	floating point value with exponent (Upper case letter E)
%g	floating point value with or without exponent
%ld	long integer
%s	String
%lf	double

Escape sequence:

- Escape sequence are special character denoted by a backslash (\) and a character after it. It is called escape sequence because it causes an 'escape' from the normal way for characters are interpreted. For example if we use '\ n', then the character is not treated as n but it is treated as a new line. Some of the common escape sequence characters are:

Escape sequence	Meaning
\a	Bell
\	Beep
\n	new line
\t	tab horizontal
\b	move the character to left one space
\\	Backslash
\'	single quote
\"	double quote