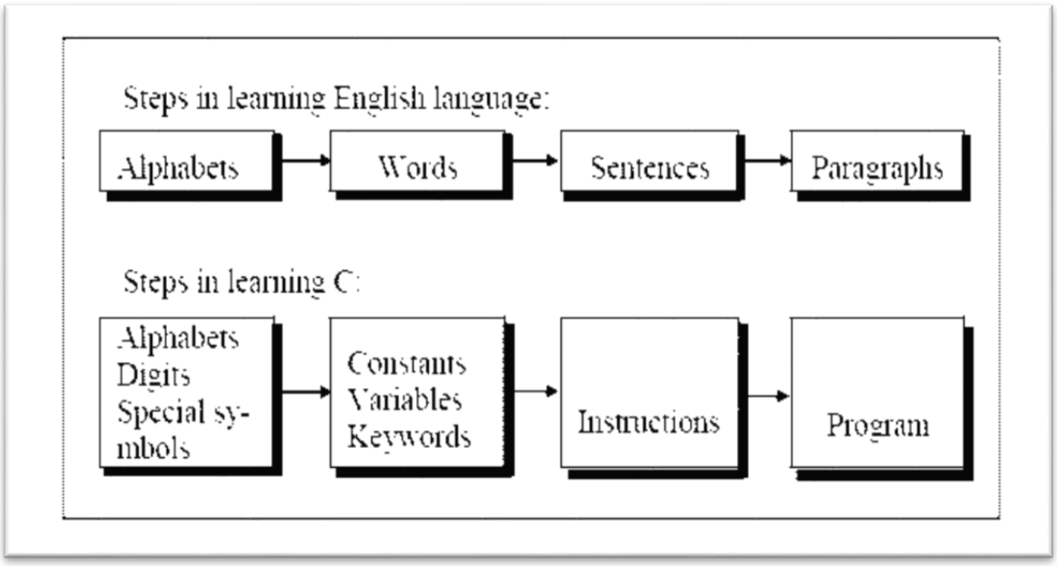
MCA First Year- KCA 102
Unit-1.3 Programming in C

Introduction

C Programing language is a general-purpose middle level (Partially low and Partially High level) Language. C is a programming language. C is one of the oldest and finest programming languages. was developed by Dennis Ritchie in 1972. It is relatively small language as its small, unambitious feature set is a real advantage: there's less to learn;

- **C programming language** is a general-purpose, procedural, structured and imperative computer programming language
- It is developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system.
- C Language is the most widely used computer language. It keeps fluctuating at number one scale of popularity along with Java programming language, which is also equally popular and most widely used among modern software programmers.
- It is partially low and partially high-level language means it can handle low-level and high-level activities
- C was invented to write an operating system called UNIX. The UNIX OS was totally written in C. It is a successor of B language which was type less
- The language was formalized in 1988 by the American National Standard Institute (ANSI).
- Most of the state-of-the-art software have been implemented using C. It can be compiled on a variety of computer platforms



Steps in learning English language:
Alphabets
Words
Sentences
Paragraphs
Steps in learning C:
Alphabets Digits Special symbols
Constants Variables Keywords
Instructions
Program

History of C Language

In the early days, in general every language was designed for some specific purpose. like FORTRAN (Formula Translator) was used for scientific and mathematical applications, COBOL (Common Business Oriented Language) was used for business applications. Thus need of a language was recognized which could withstand most of the purposes. "Necessity is the mother of invention". This was the first step towards C was put forward by Dennis Ritchie.

The C programming language was developed in 1972, at Bell laboratories by Dennis Ritchie. Initially it was designed for programming in the operating system called UNIX. After the advent of C, the whole UNIX operating system rewritten using C language only. Now almost tlvc entire UNIX operating system and the tools supplied with it including the C compiler itself are written in C. The C language is derived from the B language, which was written by Ken Thompson at AT&T Bell laboratories.

The B language was adopted from a language **called BCPL (Basic Combined Programming Language**), which was developed by Martin Richards at Cambridge University

Characteristics of C

- It is a middle level language. It has the simplicity of a high-level language as well as the power of low level language. This aspect of C makes it suitable~ for writing both application' programs and system programs.
- It is an excellent, efficient and general-purpose structured, procedural, modular imperative programming language.

- It used for mathematical, scientific, business and system software applications. C is small language, consisting of only 32 English words known as keywords (if, else, for, break etc.:
- The power of C augmented by the library functions provided with it. Moreover, the language is extendible since it allows the users to add their own library functions to the library. C contains control constructs needed to write a structured program hence it is also known as a structured programming language.
- It includes structures for selection (if.. .else, switch), repetition (while, fo do...while) and for loop exit\ (break). . .
- The programs written in C are portable i.e. programs written for one type of computer or operating system can be run\on another type of computer or operating system
- C also used to develop games, an area where latency is very important i.e. computer has to react quickly on user input.

C provides:

- Efficiency, high performance and high quality s/ws
- flexibility and power
- many high-level and low-level operations → middle level
- Stability and small size code
- Provide functionality through rich set of function libraries
- Gateway for other professional languages like C → C++ → Java

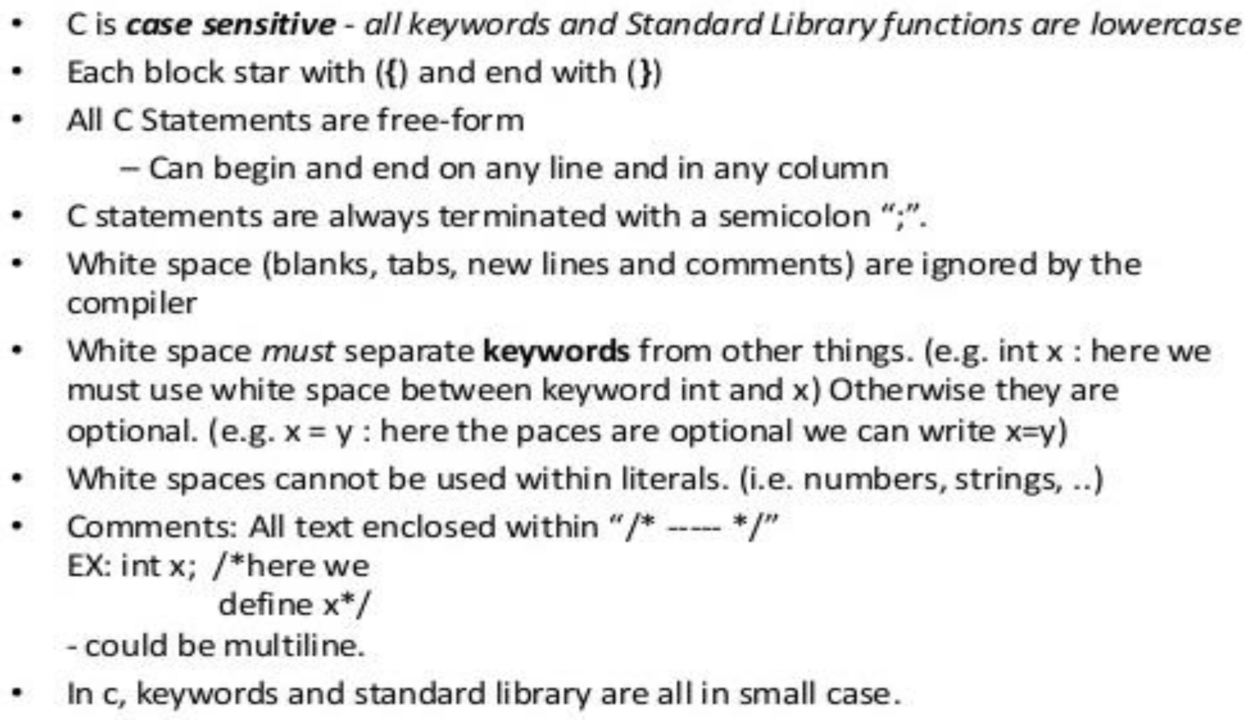
C is used:

- System software Compilers, Editors, embedded systems
- data compression, graphics and computational geometry, utility programs
- databases, operating systems, device drivers, system level routines
- there are zillions of lines of C legacy code
- Also used in application programs

Uses of C: C is a language that is used to program a wide variety of systems. Some of the uses of C are as follows:

- Major parts of Windows, Linux, and other operating systems are written in C.
- C is used to write driver programs for devices like Tablets, Printers, etc.
- C language is used to program embedded systems where programs need to run faster in limited memory.

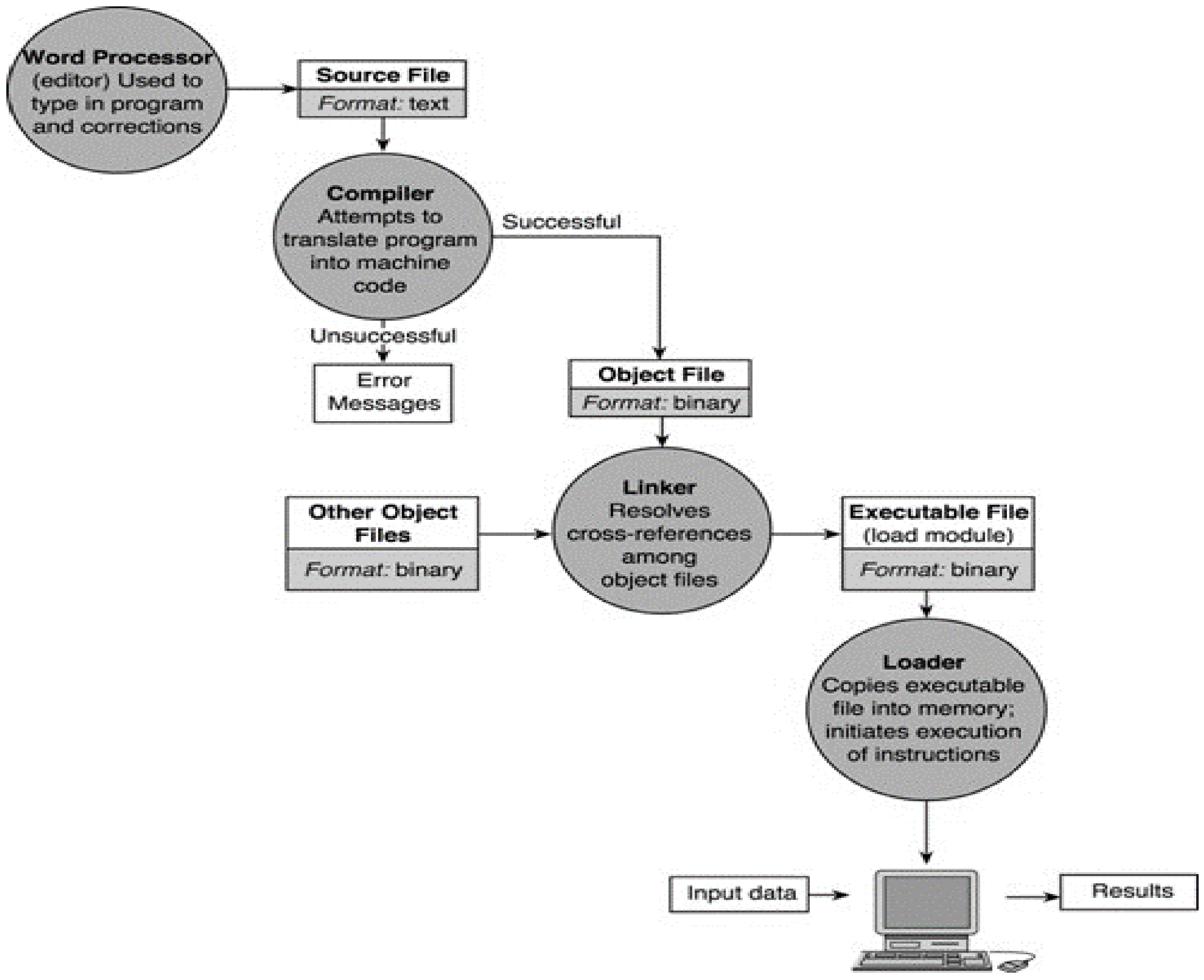
General Rules for programming in C



- C is ***case sensitive*** *- all keywords and Standard Library functions are lowercase*
- Each block star with ({) and end with (})
- All C Statements are free-form
  – Can begin and end on any line and in any column
- C statements are always terminated with a semicolon ";".
- White space (blanks, tabs, new lines and comments) are ignored by the compiler
- White space *must* separate **keywords** from other things. (e.g. int x : here we must use white space between keyword int and x) Otherwise they are optional. (e.g. x = y : here the paces are optional we can write x=y)
- White spaces cannot be used within literals. (i.e. numbers, strings, ..)
- Comments: All text enclosed within "/* ----- */"
  EX: int x; /*here we
  define x*/
  - could be multiline.
- In c, keywords and standard library are all in small case.

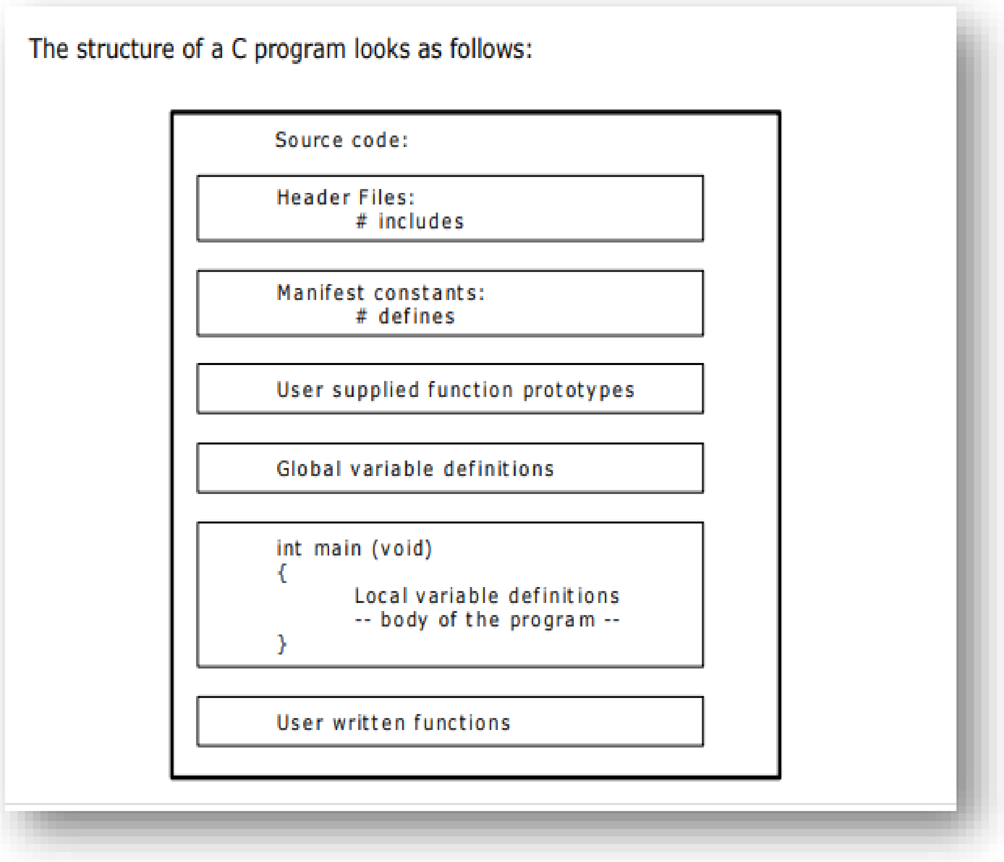
Program Development Life Cycle

Source Code: Code written with .c extension file name
Compiler: Compiler creates object code and stores it on disk. The compiler is a special type of translator. It translates source code into machine language code (same as object code). Errors: Eventually compiler errors become no big deal. You leave off a semicolon, misspell, forget a parenthesis, mess up your curly braces, etc.
Linker: Linker links the object code with the libraries, creates an executable file and stores it on disk. When you refer to something in a library, the compiler leaves a hole and the linker must resolve the addressing; in other words, find it to create a complete executable file. Errors: If you get a linking error, it can’t find something. Usually it’s a typo. Function signatures (the header) etc., must be identical. Loader: Loader puts the program in memory


Word Processor (editor) Used to type in program and corrections
Source File
Format: text
Compiler Attempts to translate program into machine code
Successful
Unsuccessful
Error Messages
Object File
Format: binary
Other Object Files
Format: binary
Linker Resolves cross-references among object files
Executable File (load module)
Format: binary
Loader Copies executable file into memory; initiates execution of instructions
Input data
Results

Structure of a C program

- In C, comments begin with the sequence /* and are terminated by */.
- Anything that is between the beginning and ending comments symbols is ignored by the compiler. Example: /* sample program */
- n C, blank lines are permitted and have no effect on the program.
- A non-trivial C application typically has following general portions on it: The structure of a C program looks as follows:


The structure of a C program looks as follows:
Source code:
Header Files:
# includes
Manifest constants:
# defines
User supplied function prototypes
Global variable definitions
int main (void)
{
Local variable definitions
-- body of the program --
}
User written functions

All c programs have to follow a basic structure. A c program starts with the main function and executes instructions presents inside it. Each instruction terminated with a semicolon(;)
There are some basic rules which are applicable to all the c programs:

1. Every program's execution starts from the main function.
2. All the statements are terminated with a semi-colon.
3. Instructions are case sensitive.
4. Instructions are executed in the same order in which they are written.

Comments
Comments are used to clarify something about the program in plain language. It is a way for us to add notes to our program. There are two types of comments in c:

5. Single line comment: //This is a comment.
6. Multi line comment : /*This is multi line comment*/

Comments in a C program are not executed and ignored.
Compilation and execution
A compiler is a computer program which converts a c program into machine language so that it can be easily understood by the computer
A program is written in plain text. This plain text is a combination of instructions in a particular sequence. The compiler performs some basic checks and finally convert the program into an executable.
Library functions

Elements of a typical C Program

Preprocessor:

C libraries : Most of its intelligence is compartmentalized in libraries Almost all c programs use the "stdio" or standard input/output library. Many also use the "math" library.

To use a library, include the header file (I.e., "stdio.h") at the top of the file.

For most special purpose libraries (I.e., math) you need to include the library on the link line.

Every C program line begins with # provides an instruction to the C preprocessor. It is executed before the actual compilation is done.

Two most common directives:

- #include
- #define

In our example (#include<stdio.h>) identifies the *header* file for standard input and output needed by the printf().

- **# includes:** An include directive tells the preprocessor to include the contents of the specified file at the point in the program. Path names must either be enclosed by double quotes or angle brackets.
- **Header Files (.h):** Header files contains declaration information for standard library functions or constants that are referred in programs. They are used to keep source-file size to a minimum and to reduce the amount of redundant information that must be coded. A header file can be included in following ways:

  1: # include <stdio.h>:
  the <> tells the preprocessor to search for the included file in a special known \include directory or directories.

  2: # include "mylib.h"
  the double quotes (" ") indicate that the current directory should be checked for the header file first. If it is not found, the special directory (or directories) should be checked.

  3 # include "mine\include\mylib.h"
  It is similar to 2, but the named relative directory \mine\include is checked for the header file mylib.h. Relative paths can also be proceeded by the .\ or ..\ notation; absolute paths always begin with a \.

  # defines: ANSI C allows you to declare constants. The # define directive is used to tell the preprocessor to perform a search-and-replace operation.

  Example:
  # define Pi 3.14159
  # define Tax-rate 0.0735

  In the example above, the preprocessor will search through the source file and replace every instance of the token Pi with 3.14159 After performing the search and replace operation, the preprocessor removes the # define line.

- **User defined function prototypes:** Declares the user-written functions actually defined later in the source file.
- **A function prototype** is a statement (rather than an entire function declaration) that is placed a head of a calling function in the source code to define the function's label before it is used. A function prototype is simply a reproduction of a function header (the first line of a function declaration) that is terminated with a semi-colon.
- **Function main () in C :**
- Compiler identify the start of the program. Every C program has a main ( ). 'main' is a C ***keyword.*** We **must not** use it for any other variable.

  There are following four common ways of main declaration

```
int
main(void)
{
   return 0;
}
```

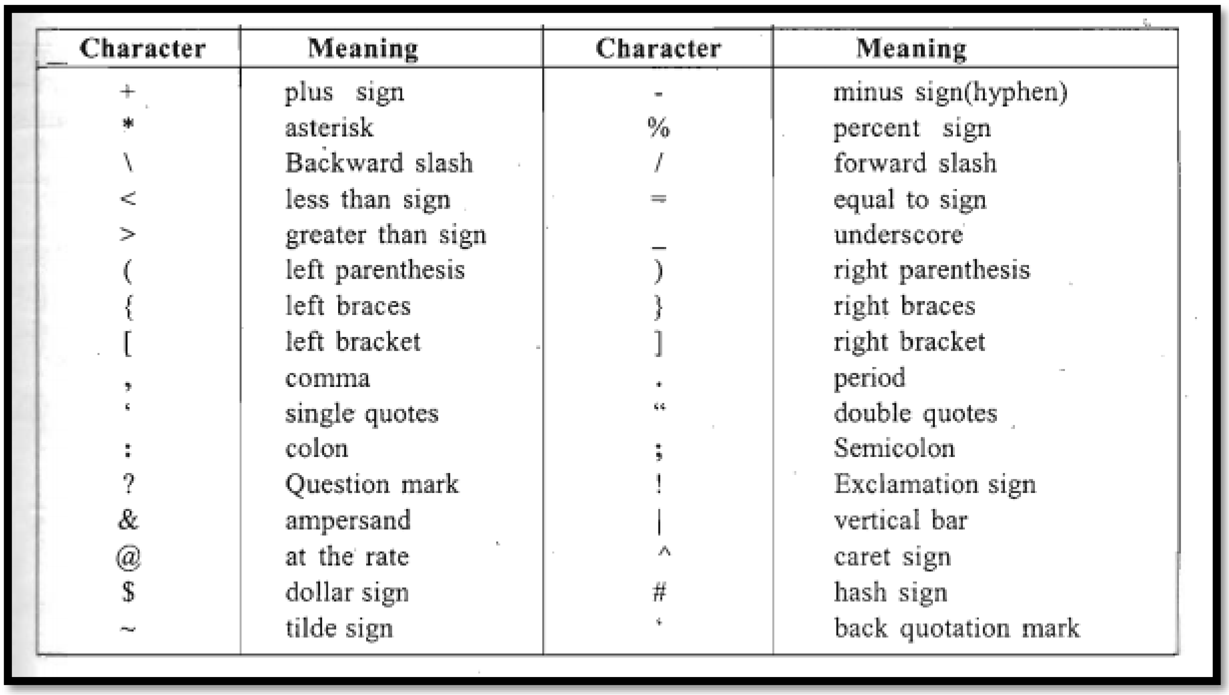
```
void
main(void)
{
}
```

```
main(void)
{
}
```

```
main( )
{
}
```

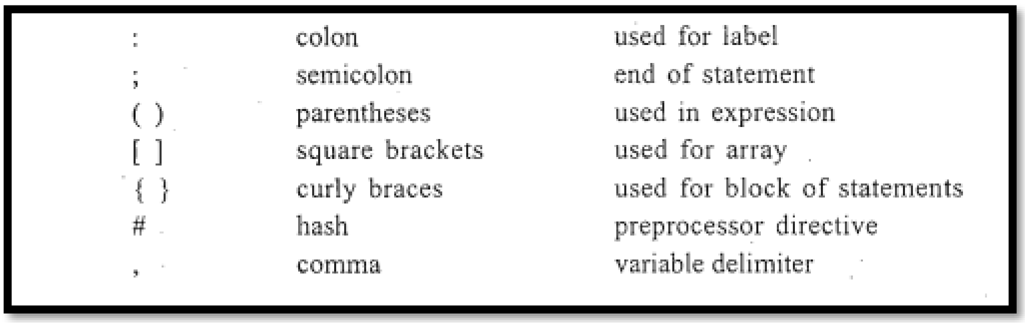
Building Blocks of C Programming Language:
C language consist of some characters set, numbers &
some special symbols. The character set of C consist of all the alphabets of English language. C consist of

1. **Character Set**
2. **Alphabets: A to Z and a to z**
3. **Numbers 0 to 9**
4. **Special characters**



Character	Meaning	Character	Meaning
+	plus sign	-	minus sign(hyphen)
*	asterisk	%	percent sign
\	Backward slash	/	forward slash
<	less than sign	=	equal to sign
>	greater than sign	_	underscore
(	left parenthesis	)	right parenthesis
{	left braces	}	right braces
[	left bracket	]	right bracket
,	comma	.	period
‘	single quotes	“	double quotes
:	colon	;	Semicolon
?	Question mark	!	Exclamation sign
&	ampersand	|	vertical bar
@	at the rate	^	caret sign
$	dollar sign	#	hash sign
~	tilde sign	‘	back quotation mark

Delimiters



:	colon	used for label
;	semicolon	end of statement
( )	parentheses	used in expression
[ ]	square brackets	used for array
{ }	curly braces	used for block of statements
#	hash	preprocessor directive
,	comma	variable delimiter

C Tokens

The words formed from the character set are building blocks of C and are sometimes known as tokens. These tokens represent the individual entity of language. Following are the different types of token used in C

1)Keywords 2) Identifiers
3)Constants 4) Operators
5)Punctuation Symbols

Keywords
These are reserved words, whose meaning is already known to the compiler. There are 32 keywords available in c:

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

Identifiers
A 'C' program consist of two types of elements, user defined and system defined. Identifiers is nothing but a name given to these elements. An identifier is a word used by a programmer to name a variable, function, or label. identifiers consist of letters and digits, in any order, except that the first character or label. Identifiers consist of letters and digits if any order, except that the first character must be letter. Both Upper and lowercase letters can be used

Rules	Example
Can contain a mix of characters and numbers. However it **cannot** start with a number	H2o
First character must be a letter or underscore	Number1; _area
Can be of mixed cases including underscore character	XsquAre my_num
Cannot contain any arithmetic operators	R*S+T
… or any other punctuation marks…	#@x%!!
Cannot be a C keyword/reserved word	struct; printf;
Cannot contain a space	My height
… identifiers are **case sensitive**	Tax != tax

- **Variables**
  - "Variables" are simply storage locations to hold data. For every variable, some memory is allocated. The
  - size of this memory depends on the type of the variable. For example, 2 bytes are allocated for integer
  - type, 4 bytes are allocated for float type, etc.
    - thus a variable is like a container that stores a 'value'. Like we have containers at home for storing rice, dal, sugar, etc. Similar to that variable in c stores value of a constant. Example:

```
a = 3;        //a is assigned "3"
b =4.7;       //b is assigned "4.7"
c='A';         //c is assigned "A"
```

Rules for naming variables in c:
1. The first character must be an alphabet or underscore (_).

2. No commas, blanks allow.
3. No special symbol other than underscore is allowed
4. Variable names are case sensitive

Escape Sequences

Sometimes, it is necessary to use characters which cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc. In order to use these characters, escape sequence is used.

For example: \n is used for newline. The backslash (\ ) causes "escape" from the normal way the characters are interpreted by the compiler. Escape

Sequences	Character
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null character

C language has a lot of valuable library functions which is used to carry out a certain task, for instance, printf function is used to print values on the screen.

```
printf("This is %d",i);
        // %d for integers
        // %f for real values
        // %c for characters
```

Types of variables

Integer variables - int a=3;
Real variables - int a=7.7 (wrong as 7.7 is real) ; float a=7.7;
Character variables - char a='B';
C Data Types

In the C programming language, data types refer to an extensive system used for declaring variables or functions of different types.

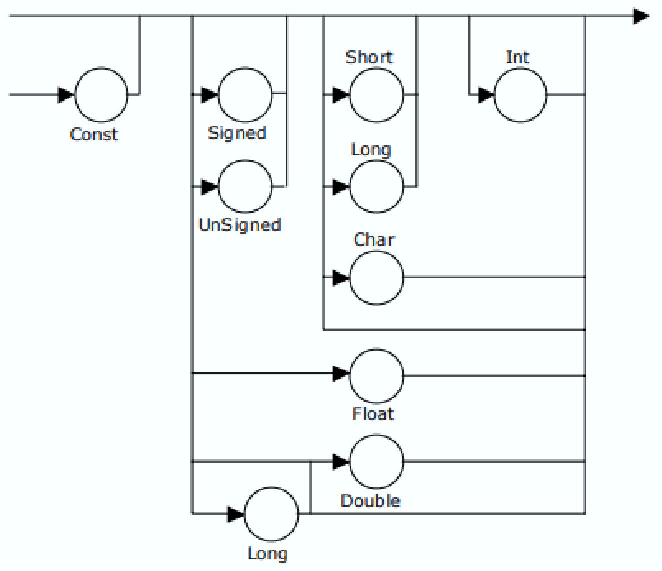
The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted. The types in C can be classified as follows: S.N. Types and Description

1 Basic Types: They are arithmetic types and consists of the two types: (a) integer types and (b) floatingpoint types

C has the following simple data types (16-bit implementation):

Type	**Size**	**Range**	**Precision for real numbers**
char	1 byte	-128 to 127	
unsigned char	1 byte	0 to 255	
signed char	1 byte	-128 to 127	
short int or short	2 bytes	-32,768 to 32,767	
unsigned short or unsigned short int	2 bytes	0 to 65535	
int	2 bytes	-32,768 to 32,767	
unsigned int	2 bytes	0 to 65535	
Long or long int	4 bytes	-2147483648 to 2147483647 (2.1 billion)	
unsigned long or unsigned long int	4 bytes	0 to 4294967295	
float	4 bytes	3.4 E-38 to 3.4 E+38	6 digits of precision
double	8 bytes	1.7 E-308 to 1.7 E+308	15 digits of precision
long double	10 bytes	+3.4 E-4932 to 1.1 E+4932	provides between 16 and 30 decimal places

The syntax of the simple type specifier:



Refernces

1. . Hanly J. R. and Koffman E. B.,“Problem Solving and Program Design in C”, Pearson Education.
2. Schildt H., “C- The Complete Reference”, McGraw-Hill.
3. Kanetkar Y., “Let Us C”, BPB Publications.
4. Gottfried B., “Schaum’s Outlines- Programming in C”, McGraw-Hill Publications.
5. Kochan S.G., “Programming in C”, Addison-Wesley.
6. Dey P. and Ghosh M., “Computer Fundamentals and Programming in C”, Oxford University Press.
7. Goyal K. K., Sharma M. K. and Thapliyal M. P. “Concept of Computer and C Programming”, University Science Press