UNIT – 2 Elements of C

Every computer programming language is a set of syntax rules which can be formally stated as technical specifications. However, many languages like C are created under some specific historic settings in which standardization of the language is not the most important thing at its creation time. As time goes by, this causes big problems since the lack of standardization lays the foundation of incompatible language dialects. Also, the language needs to be improved by adding new language features and syntactic structures. This could only be achieved by updating a pre-exist standard so that every compiler vendor knows how to implement the new features. So that's why we have C standards.

The **standard called K&R** C was actually an informal specification based on the first edition of the extremely popular book, 'The C Programming Language', published in 1978 by Brian Kernighan and Dennis Ritchie; hence, the name K&R after Kernighan and Ritchie

K&R C not only acted as an informal standards specification for C but also added language features like the new data types *long int* and *unsigned int* and the compound assignment operator. A standardized I/O library was also proposed by K&R C.

ANSI C

Even though K&R C was accepted by many programmers as the basic standard of C, it was not the Complete standard, and nobody could have been persuaded into accepting it as the official standard of C.
So, it was absolutely essential for some standards organization to accept the challenge of coming up with an official standard for C. The American National Standards Institute (ANSI) addressed this issue in 1983 by forming a committee, and the final draft of the standards it formulated was released in 1989. This is the reason why ANSI C is also called C89.

C99

C99 is the informal name given to the ISO/IEC 9899:1999 standards specification for C that was adopted in 1999. The C99 standard added five more keywords to ANSI C, and the total number of keywords became 37.
The keywords added in C99 are _Bool, _Complex, _Imaginary, inline and restrict. The keyword _Bool is used to declare a new integer type capable of storing 0 and 1. The keywords _Complex and _Imaginary are used to declare complex and imaginary floating point type variables to handle complex numbers. The keyword inline is used to declare inline functions in C, similar to C++ inline functions. The keyword restrict is used to tell the compiler that for the lifetime of the pointer, only the pointer itself or a value directly derived from it will be used to access the object to which it points. New header files like <stdbool.h>, <complex.h>, <tgmath.h>, <inttypes.h>, etc, were also added in C99. A new integer data type called long long int with a minimum size of 8 bytes was added in C99.

Assignment: Try to explain other C standards too

Fundamental of C

Program is a set of instructions to interact with computer. The programs are written in programing language.
C is a procedural programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie.
C program is a sequence of characters that will be converted by the C compiler into machine code.

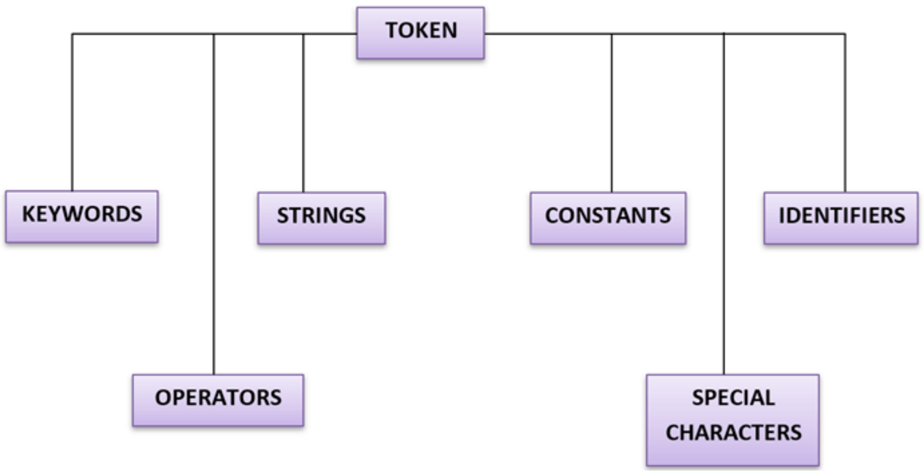
Character set of C

Character set is a package of valid characters recognized by compiler/interpreter. Each natural language has its own alphabet and characters set as the basic building. We combine different alphabets to make a word, sentence, paragraph and a meaningful passage. Similarly each computer languages have their own character set. Particularly, C also has its own character set, shows in following table:

Letters	A, B, C…………………Z
	a, b, c……………………z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	+ - * / ! ? | \ < > & , : ; # % * ( ) [ ] . _ $ ^
White space characters	Blank space, new line, horizontal tab, vertical tab, etc

Token

A token is the smallest unit in a 'C' program. A token is divided into six different types as follows



1. Keywords

Keywords are the predefined reserved (built-in) words whose meaning has already been defined to the c compiler. A programmer can't use the keywords for another task than the task for which it has been defined. For example: a programmer can't use the keyword for variable names.

There are 32 keywords in c, which are listed in figure:

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

2. Identifiers

Identifiers are the user defined names and consists of a sequence of letters and digits with letter or underscore as a first character. It may be name of the variables, functions, arrays, etc. In identifiers we can use both upper and lower cases.

Rules for Identifiers

- First characters must be an alphabet or underscore.
- Must consist of only alphabets, digits or underscore.

- Cannot use a keyword.
- Must not contain white spaces. To connect two words underscore can be used.

Examples for valid or invalid identifier

a. record1

→This is valid identifier.

b. file_3

→This is valid identifier.

c. #tax

→Invalid identifier because identifier must begin with a letter and special character like # is not permitted in identifiers.

d. goto

→Invalid identifier because keywords cannot be used as an identifier.

e. Name-and-address

→Invalid identifier because hyphen or dash is not permitted.

f. void

→Invalid identifier because keywords cannot be used as an identifier

3. Constants

Constant in C refers to fixed values that do not change during execution of a program. C supports several types of constants, such as numeric constants and character constants.

- **Numeric constants:**

  It consists of:

  Integer Constants:

  It refers to the sequence of digits with no decimal points. The three kinds of integer constants are:

  - **Decimals**: Decimal numbers are set of digits from 0 to 9 with leading +ve or –ve sign. For example: 121, -512 etc.
  - **Octals**: Octal numbers are any combination of digits from set 0 to 7 with leading 0. For example: 0121, 0375 etc.
  - **Hexadecimals**: Hexadecimal numbers are the sequence of digits 0-9 and alphabets a-f (A-F) with preceding 0X or 0x. For example: 0xAB23, 0X787 etc.

  Real or Floating point Constants:

  They are the numeric constants with decimal points. The real constants are further categorized as:

  - **Fractional Real constant:** They are the set of digits from 0 to 9 with decimal points. For example: 394.7867, 0.78676 etc.
  - **Exponential Real constants:** In the exponential form, the constants are represented in two form: Mentissia E exponent;
    Where, the Mentissia is either integer or real constant but the exponent is always in integer form.
    For
    example: 21565.32 = 2.156532 E4, where E4 = 10^4.

- **Character constant:**

  Character constant are the set of alphabets. Every character has some integer value known as American Standard Code for Information Interchange (ASCII). The character constants are further categories as:

  Single character constant:

  It contains a single character enclosed with in a pair of single quote mark (‘ ’). Thus, ‘X’, ‘5’ are characters but X, 5 are not.

  String constant:

  String constants are a sequence of characters enclosed in double quotes marks (“ ”). They may be letters, numbers, special symbols or blank spaces. For example: “Hello”, “X+Y”. Note: ‘x’ ≠ “x”

Characters	ASCII Values
A - Z	(65 – 90)
a - z	(97 – 122)
0 - 9	(48 – 57)

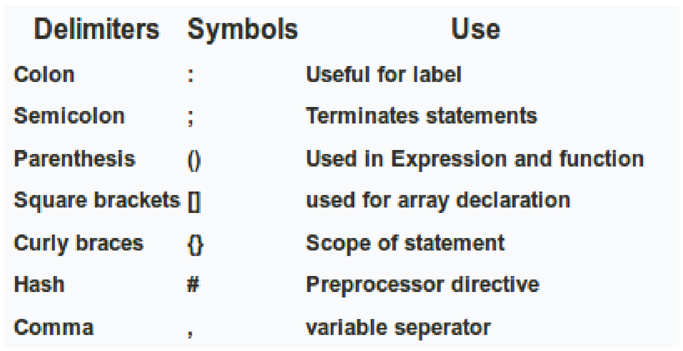
4. **Delimiters**

Delimiters are small system components that separate the various elements (variables, constants, statements) of a program. They are also known as separators.

Some of the delimiters are:

- Comma: It is used to separate two or more variables, constants.
- Semicolon: It is used to indicate the end of a statement.
- Apostrophes (single quote): It is used to indicate character constant.
- Double quotes: It is used to indicate a string.
- White space: Space, tab, new line etc.

```
#include<stdio.h>
void main()
{
        int a,b,c[10];
        label_num:
        printf("hi");
        goto label_num:
}
```



Delimiters	Symbols	Use
Colon	:	Useful for label
Semicolon	;	Terminates statements
Parenthesis	()	Used in Expression and function
Square brackets	[]	used for array declaration
Curly braces	{}	Scope of statement
Hash	#	Preprocessor directive
Comma	,	variable seperator

5. **Operators**

C supports a rich set of operators. An operator is a symbol that tells the computer to perform mathematical or logical operations on operands. Operators are used in programs to manipulate data.

Variables

A variable is a data name that may be used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution. Constant supplied by the user is stored in block of memory by the help of variable.

Consider an expression:

$2*x + 3*y = 5.$

In this expression the quantities x and y are variable. A variable is a named location in memory that is used to hold certain values.

- Rules for declaring variable name

1. Every variable name in C must start with a letter or underscore.
2. A valid variable name should not contain commas or blanks.
3. Valid variable name should not be keyword.
4. A variable name must be declared before using it.

-Declaring variable

After defining suitable variable name, we must declare them before used in our program.

The declaration deals with two things:

- It tells the compiler what the variable name is.

- It specifies what type of data the variable can hold.

Syntax:

Data_type *variable_names*;

For example:

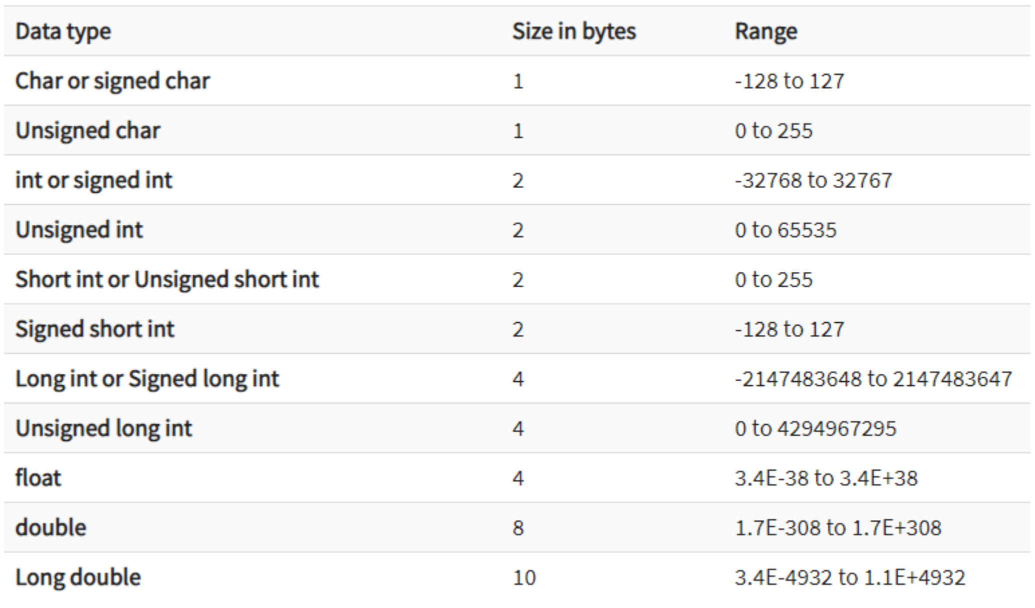
```
 int  account_number;
float  a,  b,  c;
char  name[10];
```

Data Types

Data types are used to declare the variables and tell the compiler what types of data the variable will hold. Data type is useful in C that tells the computer about the type and nature of the value to be stored in a variable. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted. Generally, the data types can be categorized as:

1. Primary Data Types

These are the basic data types. There are four basic data types in C (Integer, Character, Float and double)
They are integral data type(integer and constant fall under this type), real or floating data type and void type.



Data type	Size in bytes	Range
Char or signed char	1	-128 to 127
Unsigned char	1	0 to 255
int or signed int	2	-32768 to 32767
Unsigned int	2	0 to 65535
Short int or Unsigned short int	2	0 to 255
Signed short int	2	-128 to 127
Long int or Signed long int	4	-2147483648 to 2147483647
Unsigned long int	4	0 to 4294967295
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
Long double	10	3.4E-4932 to 1.1E+4932

2. Derived (or Secondary) Data Types

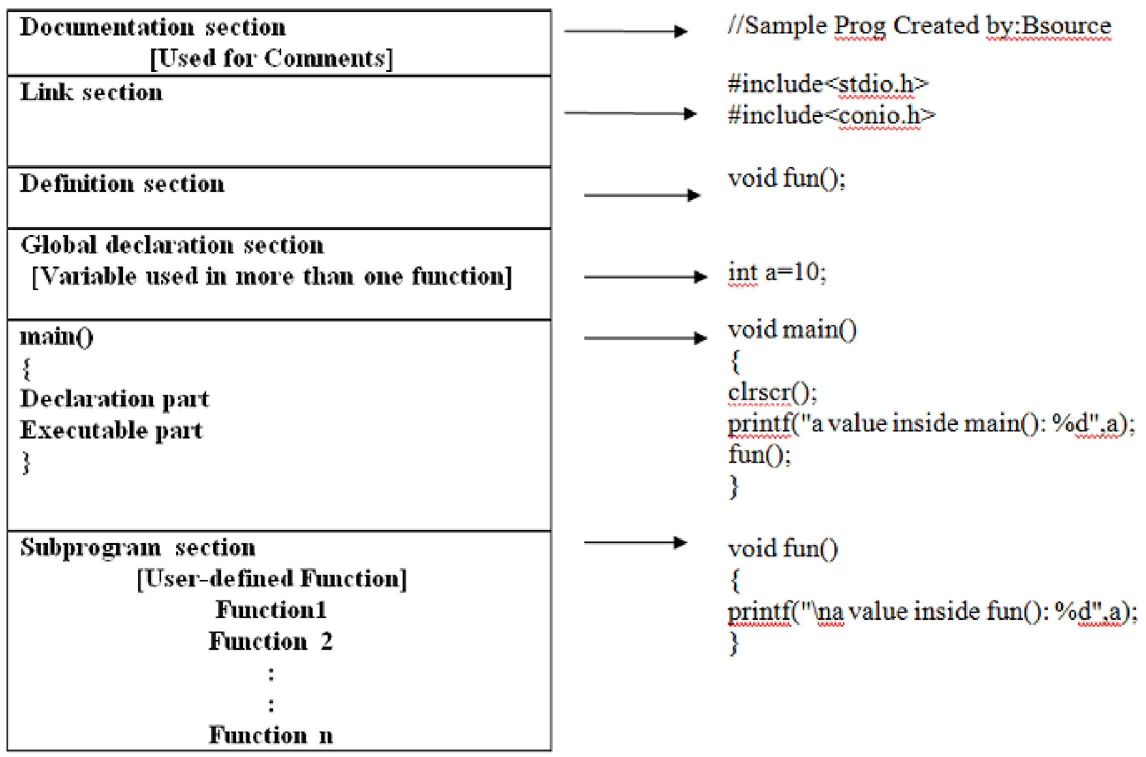
The secondary (derived) data types are a collection of primary (primitive) data types. These are derived from the primary data types. Example: array, union, structure, etc

3. User Defined Data Types

The data types defined by the user is known as user defined data types. Example: enum(enumerated data type) and typedef (type definition data type).

Basic Structure of a C Program:

BASIC STRUCTURE OF A 'C' PROGRAM: **Example:**


Documentation section [Used for Comments]
Link section
Definition section
Global declaration section [Variable used in more than one function]
main() { Declaration part Executable part }
Subprogram section [User-defined Function] Function1 Function 2 : : Function n

```
//Sample Prog Created by:Bsource

#include<stdio.h>
#include<conio.h>

void fun();

int a=10;

void main()
{
clrscr();
printf("a value inside main(): %d",a);
fun();
}

void fun()
{
printf("\na value inside fun(): %d",a);
}
```

- Documentation section :
  The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.
- Link section :
  The link section provides instructions to the compiler to link functions from the system library.
- Definition section :
  The definition section defines all symbolic constants.
- Global declaration section :
  There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.
- main () function section :
  Every C program must have one main function section. This section contains two parts; declaration part and executable part
  - Declaration part :
    The declaration part declares all the variables used in the executable part.
  - Executable part :
    There is at least one statement 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 closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.
- Subprogram section :
  The subprogram section contains all the user-defined functions that are called in the main () function. User-defined functions are generally placed immediately after the main () function, although they may appear in any order. Note:All section, except the main () function section may be absent when they are not required.
- Sample C Program:

```
#include<stdio.h> ←---------------------------------Preprocessing Directive
void main()
```

```
{ ←------------------------------------------------------Start of a Program

    /*…………Printing Starts…………*/

    Printf("Learn at every moment");

    /*…………Printing stops……..*/

} ←---------------------------------------------------------End of a Program
```

In C many library functions are grouped category-wise and stored in different files known as header files.
Ex. stdio.h

Statement

A statement is a command given to the computer that instructs the computer to take a specific action, such as display to the screen, or collect input.
A computer program is made up of a series of statements.
Example:

a. Iteration Statements (for, while etc)
b. Selection Statements (if, if then etc)
c. Jump Statements (goto etc)