C character Set

The character set in C Language can be grouped into the following categories.

1. Letters
2. Digits
3. Special Characters
4. White Spaces

You can use any of the following characters from the ASCII character set to enter programming text into your source file.

- a b c d e f g h i j k l m n o p q r s t u v w x y z
- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- 0 1 2 3 4 5 6 7 8 9
- ! " # % & ' ( ) * + , - . / :
- ; < = > ? [\ ] ^ _ { | } ~
- The space character.
- The control characters representing horizontal tab, vertical tab, carriage return, new line and form feed.

Keywords and Identifiers

Every word in C language is a keyword or an identifier. Keywords in C language cannot be used as a variable name. They are specifically used by the compiler for its own purpose and they serve as building blocks of a c program.

The following are the Keyword set of C language.

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

Some compilers may have additional keywords listed in C manual.

Identifiers refer to the name of user-defined variables, array and functions. A variable should be essentially a sequence of letters and or digits and the variable name should begin with a character.
Both uppercase and lowercase letters are permitted. The underscore character is also permitted in identifiers.

The identifiers must conform to the following rules.

1. First character must be an alphabet (or underscore)
2. Identifier names must consists of only letters, digits and underscore.
3. An identifier name should have less than 31 characters.

4. Any standard C language keyword cannot be used as a variable name.
5. An identifier should not contain a space.

Constants

A constant value is the one which does not change during the execution of a program. C supports several types of constants.

1. Integer Constants
2. Real Constants
3. Single Character Constants
4. String Constants

Integer Constants

An integer constant is a sequence of digits. There are 3 types of integers namely decimal integer, octal integers and hexadecimal integer.

Decimal Integers consists of a set of digits 0 to 9 preceded by an optional + or - sign. Spaces, commas and non digit characters are not permitted between digits. Example for valid decimal integer constants are

123
-31
0
562321
+ 78

Some examples for invalid integer constants are

15 750
20,000
Rs. 1000

Octal Integers constant consists of any combination of digits from 0 through 7 with a O at the beginning. Some examples of octal integers are

O26
O
O347
O676

Hexadecimal integer constant is preceded by 0X or 0x, they may contain alphabets from A to F or a to f. The alphabet A to F refers to 10 to 15 in decimal digits. Examples of valid hexadecimal integers are

0X2
0X8C
0Xbcd
0x

Real Constants

Real Constants consists of a fractional part in their representation. Integer constants are inadequate to represent quantities that vary continuously. These quantities are represented by numbers containing fractional parts like 26.082. Examples of real constants are

0.0026
-0.97

435.29
+487.0
Real Numbers can also be represented by exponential notation. The general form for exponential notation is mantissa exponent. The mantissa is either a real number expressed in decimal notation or an integer. The exponent is an integer number with an optional plus or minus sign.

Single Character Constants
A Single Character constant represent a single character which is enclosed in a pair of quotation symbols.
Example for character constants are
'5'
'x'
'.'
;
All character constants have an equivalent integer value which is called ASCII Value.

String Constants
A string constant is a set of characters enclosed in double quotation marks. The characters in a string constant sequence may be a alphabet, number, special character and blank space. Example of string constants are
"VISHAL"
"1234"
"God Bless"
"!.....?"

Backslash Character Constants [Escape Sequences]
Backslash character constants are special characters used in output functions. Although they contain two characters they represent only one character. Given below is the table of escape sequence and their meanings.

Constant	Meaning
'\a'	.Audible Alert (Bell)
'\b'	.Backspace
'\f'	.Form feed
'\n'	.New Line
'\r'	.Carriage Return
'\t'	.Horizontal tab
'\v'	.Vertical Tab
'\''	.Single Quote
'\"'	.Double Quote
'\?'	.Question Mark
'\\'	.Back Slash
'\0'	.Null

Variables
A variable is a value that can change any time. It is a memory location used to store a data value. A variable name should be carefully chosen by the programmer so that its

use is reflected in a useful way in the entire program. Variable names are case sensitive. Examples of variable names are
Sun
number
Salary
Emp_name
average1

Any variable declared in a program should confirm to the following
1. They must always begin with a letter, although some systems permit underscore as the first character.
2. The length of a variable must not be more than 8 characters.
3. White space is not allowed and
4. A variable should not be a Keyword
5. It should not contain any special characters.
Examples of Invalid Variable names are
123
(area)
6th
%abc

C Programming Language - Data Types

A C language programmer has to tell the system before-hand, the type of numbers or characters he is using in his program. These are data types. There are many data types in C language. A C programmer has to use appropriate data type as per his requirement.

C language data types can be broadly classified as
Primary data type
Derived data type
User-defined data type

Primary data type
All C Compilers accept the following fundamental data types

1.	Integer	int
2.	Character	char
3.	Floating Point	float
4.	Double precision floating point	double
5.	Void	void

The size and range of each data type is given in the table below

DATA TYPE	**RANGE OF VALUES**
char	-128 to 127
Int	-32768 to +32767
float	3.4 e-38 to 3.4 e+38
double	1.7 e-308 to 1.7 e+308

Integer Type :
Integers are whole numbers with a machine dependent range of values. A good programming language as to support the programmer by giving a control on a range of numbers and storage space. C has 3 classes of integer storage namely short int, int and long int. All of these data types have signed and unsigned forms. A short int requires half the space than normal integer values. Unsigned numbers are always positive and consume all the bits for the magnitude of the number. The long and unsigned integers are used to declare a longer range of values.

Floating Point Types :
Floating point number represents a real number with 6 digits precision. Floating point numbers are denoted by the keyword float. When the accuracy of the floating point number is insufficient, we can use the double to define the number. The double is same as float but with longer precision. To extend the precision further we can use long double which consumes 80 bits of memory space.

Character Type :
A single character can be defined as a defined as a character type of data. Characters are usually stored in 8 bits of internal storage. The qualifier signed or unsigned can be explicitly applied to char. While unsigned characters have values between 0 and 255, signed characters have values from –128 to 127.

Size and Range of Data Types on 16 bit machine.

type	SIZE (Bits)	Range
Char or Signed Char	8	-128 to 127
Unsigned Char	8	0 to 255
Int or Signed int	16	-32768 to 32767
Unsigned int	16	0 to 65535
Short int or Signed short int	8	-128 to 127
Unsigned short int	8	0 to 255
Long int or signed long int	32	-2147483648 to 2147483647
Unsigned long int	32	0 to 4294967295
Float	32	3.4 e-38 to 3.4 e+38
Double	64	1.7e-308 to 1.7e+308
Long Double	80	3.4 e-4932 to 3.4 e+4932

Declaration of Variables
Every variable used in the program should be declared to the compiler. The declaration does two things.
1. Tells the compiler the variables name.
2. Specifies what type of data the variable will hold.

The general format of any declaration is

datatype v1, v2, v3, …….. vn**;**

Where v1, v2, v3 are variable names. Variables are separated by commas. A declaration statement must end with a semicolon.

Example:
int sum;
int number, salary;
double average, mean;

Datatype	**Keyword Equivalent**
Character	char
Unsigned Character	unsigned char
Signed Character	signed char
Signed Integer	signed int (or) int
Signed Short Integer	signed short int (or) short int (or) short
Signed Long Integer	signed long int (or) long int (or) long
UnSigned Integer	unsigned int (or) unsigned
UnSigned Short Integer	unsigned short int (or) unsigned short
UnSigned Long Integer	unsigned long int (or) unsigned long
Floating Point	float
Double Precision Floating Point	double
Extended Double Precision Floating Point	long double

User defined type declaration
In C language a user can define an identifier that represents an existing data type. The user defined datatype identifier can later be used to declare variables. The general syntax is

typedef type identifier;

here type represents existing data type and 'identifier' refers to the 'row' name given to the data type.

Example:

typedef int salary;
typedef float average;

Here salary symbolizes int and average symbolizes float. They can be later used to declare variables as follows:

salary dept1, dept2;
average section1, section2;

Therefore dept1 and dept2 are indirectly declared as integer datatype and section1 and section2 are indirectly float data type.

The second type of user defined data type is enumerated data type which is defined as follows.

Enum identifier {value1, value2 …. Valuen};

The identifier is a user defined enumerated data type which can be used to declare variables that have one of the values enclosed within the braces. After the definition we can declare variables to be of this 'new' type as below.

enum identifier V1, V2, V3, …… Vn

The enumerated variables V1, V2, ….. Vn can have only one of the values value1, value2 ….. valuen

Example:

```
enum day {Monday, Tuesday, …. Sunday};
enum day week_st, week_end;
week_st = Monday;
week_end = Friday;
if(wk_st = = Tuesday)
        week_en = Saturday;
```

Declaration of Storage Class

Variables in C have not only the data type but also storage class that provides information about their location and visibility. The storage class divides the portion of the program within which the variables are recognized.

auto : It is a local variable known only to the function in which it is declared. Auto is the default storage class.

static : Local variable which exists and retains its value even after the control is transferred to the calling function.

extern : Global variable known to all functions in the file.

register : Social variables which are stored in the register.

Defining Symbolic Constants

A symbolic constant value can be defined as a preprocessor statement and used in the program as any other constant value. The general form of a symbolic constant is

#define symbolic_name value_of_constant

Valid examples of constant definitions are :

```
# define marks 100
#define total 50
#define pi 3.14159
```

These values may appear anywhere in the program, but must come before it is referenced in the program. It is a standard practice to place them at the beginning of the program.

Declaring Variable as Constant

The values of some variable may be required to remain constant through-out the program. We can do this by using the qualifier const at the time of initialization.

Example:

Const int class_size = 40;

The const data type qualifier tells the compiler that the value of the int variable class_size may not be modified in the program.

Volatile Variable

A volatile variable is the one whose values may be changed at any time by some external sources.

Example:

volatile int num;

The value of data may be altered by some external factor, even if it does not appear on the left hand side of the assignment statement. When we declare a variable as volatile the compiler will examine the value of the variable each time it is encountered to see if an external factor has changed the value.