-C Charset and Variables-

CO-101 Programming Fundamentals

By Priya Singh (Assistant Professor), Dept. of S.E., DTU

C Character Set

- Like any other language, C contains a set of characters used to construct words, statements, etc.
- Supports a total of 256 characters.
- Every C program contains statements.
- These statements are constructed using words and these words are constructed using characters from C character set.
- C language character set contains the following set of characters:

1. Alphabets/Letters/Characters (a-z A-Z)
2. Digits (0-9)
3. Special Symbols (> < = ! &)

   a : value at a

   &a: address of a

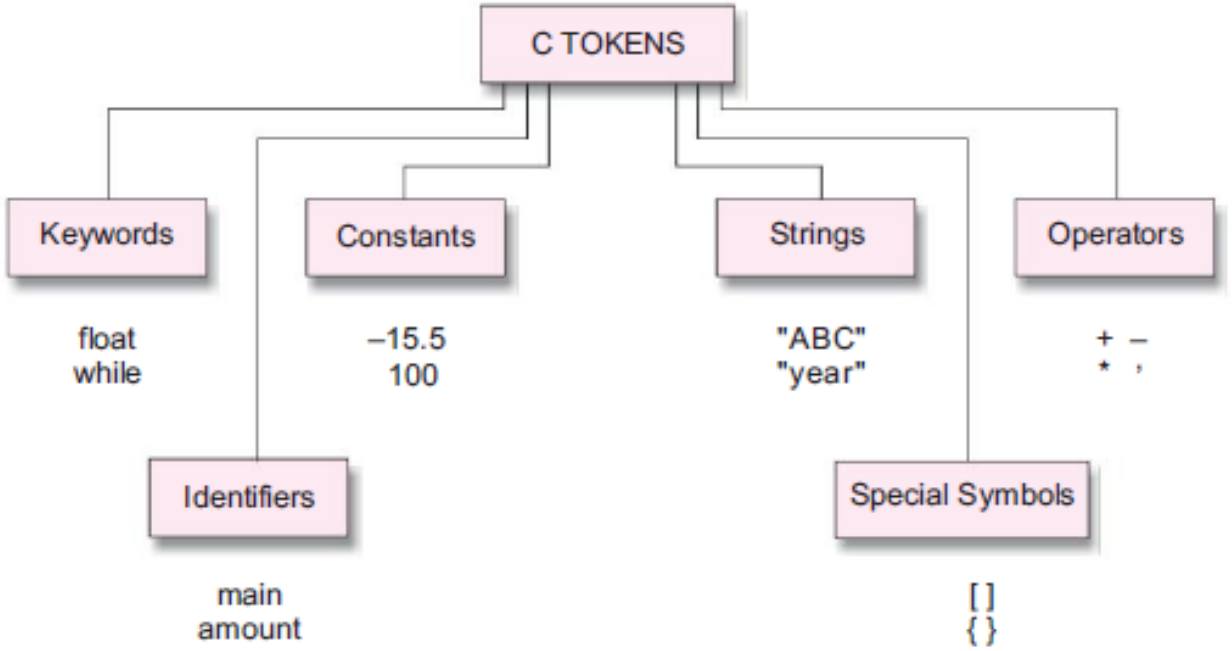
4. White spaces (blanks, tabs)

Trigraph in C:
Many non English keyboards do not support some of the characters from C character sets. C provides a way to type those using ??

Trigraph sequence	*Translation*
??=	# number sign
??(	[left bracket
??)	] right bracket
??<	{ left brace
??>	} right brace
??!	| vetical bar
??/	\ back slash
??/	^ caret
??-	~ tilde

C Tokens

- Every smallest individual unit of a c program is called token.
- Tokens are used to construct c programs and they are said to the basic building blocks of a C program.



Keywords and Identifiers

Every C word: keyword/ identifier

Keywords are the reserved words with predefined meaning which already known to the compiler

Properties of Keywords:

- They must be used only in lowercase letters
- Every keyword has a specific meaning
- Can not be used as user-defined names like variable, functions, arrays, pointers, etc.
- Represents something or specifies some kind of action to be performed by the compiler.

List of keywords in C:

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

Identifiers:

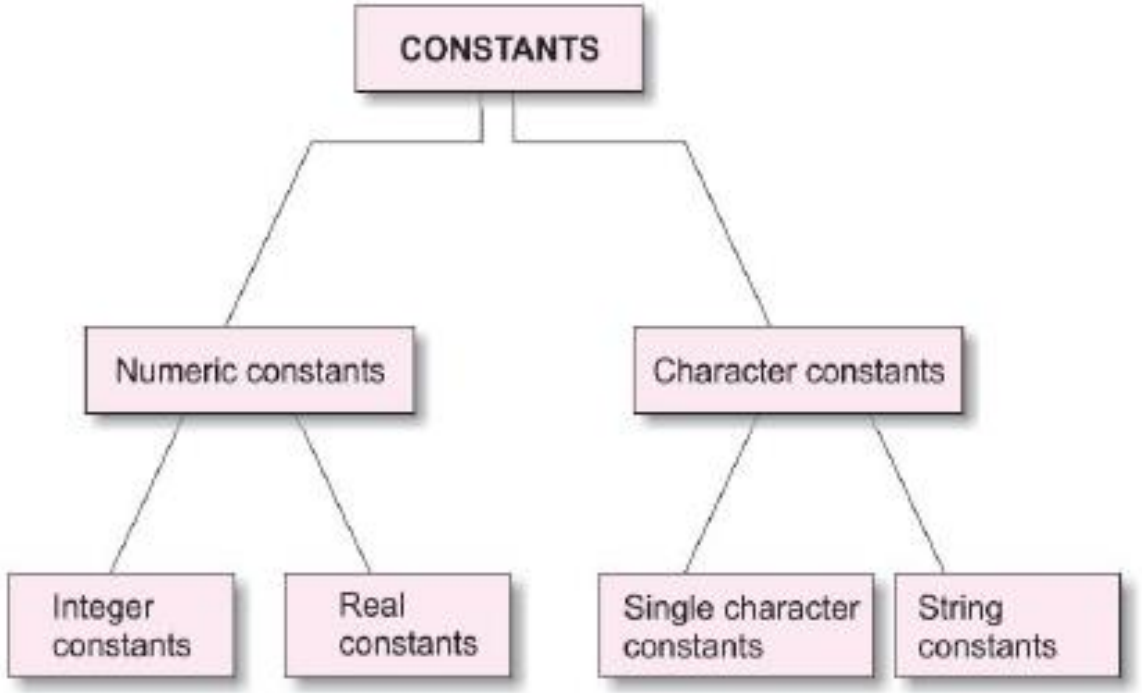
- The identifier is a user-defined name of an entity to identify it uniquely during the program execution.
- It is the name of variable, function or an array.
- In C programming language, programmers can specify their name to a variable, array, pointer, function, etc... An identifier is a collection of characters which acts as the name of variable, function, array, pointer, structure, etc.

Rules for Creating Identifiers:

- An identifier can contain letters (UPPERCASE and lowercase), numeric & underscore symbol only.
- An identifier should not start with a numerical value. It can start with a letter or an underscore.
- We should not use any special symbols in between the identifier even whitespace. However, the only underscore symbol is allowed.
- Keywords should not be used as identifiers.
- There is no limit for the length of an identifier. However, the compiler considers the first 31 characters only.
- An identifier must be unique in its scope.

Constants

- Constants in C refer to fixed values that do not change during execution of



Integer Constants:

- Embedded spaces, commas, and non-digit characters are not allowed b/w digits.
- Octal constants: 0 followed by any combination of 0-7(2 power 3=>8)
- Hexadecimal constants: 0x followed by any combination of 0-9 and A-F
- In programming octal and hexadecimal are rarely used.

Real Constants:

- These are shown by whole number followed by decimal symbol ‘.’ and the fractional part. E.g.: 123.45
- Can also be shown as exponential/scientific notation. E.g.: 1.2345e2
- (e2: multiples of 10 power 2)
- Embedded white space is not allowed.

Single Character Constants:

It contains a single character within single quotes.

Character (valid ASCII character values) constants have integer value – ASCII

e.g.:

printf(“%d”,’a’); => 97 ascii

printf(“%c”,’97’); => a

a: 97
z: 26th: 97+25=122

A: 65
Z: 65+25=90

1: 49

String constants/literals:

Sequence of characters enclosed in double quotes.

Characters can be letters, numbers, special characters, blank spaces.

Note: A character constant is not same as single character string.

e.g.: ‘a’ (character constant) and “a” (string constant with only one character) are not equivalent.

Also, a single character string does not have integer equivalent value unlike a character value.

Backslash character constant: \n \t \0 \\

Used in the O/P function

Each of the following backslash characters represent one character.

Constant	*Meaning*
'\a'	audible alert (bell)
'\b'	back space
'\f'	form feed
'\n'	new line
'\r'	carriage return
'\t'	horizontal tab
'\v'	vertical tab
'\''	single quote
'\"'	double quote
'\?'	question mark
'\\'	backslash
'10'	null

Variable

- Variable is a name given to a memory location where we can store different values of the same datatype during the program execution.

1 Byte=> 8 bits

- Every program instruction must confirm precisely to syntax rules of language –compiler.
- Declaring Data Variables: In C all variables are declared before they are used.

- int a;
- a=45;

a: 1000

- a: value inside a: 45
- &a: address of a: 1000

This is so that:

1. A memory location is given a name.
2. A suitable number of bytes can be allocated.
3. The compiler knows how to treat the data

- The char, int, short and long types can be preceded by the qualifiers signed or unsigned.
- The default is signed.
- If used on their own the type **signed** refers to type signed int and **unsigned** refers to type unsigned int.
- The type char is so called as it is suitable for storing a character but the 'C' compiler will also let it be used to store numbers.
- Similarly int, short or long variables, either signed or unsigned may be used for storing characters.
- The number of bits for each type will vary from one compiler to the next, even on the same type of computer.

- Each variable declaration statement consists of a type name followed by one or more variable names.
- There is no limit to the number or order of variable declarations.

- Variable names must obey the following rules:

1. Names can consist of letters, digits, "_"
2. Names must start with a letter
3. Names can start with the "_", underscore character but this is not recommended as many system macros and functions are given names in this format.
4. Case is significant, ie. Xyz is not the same as xyz
5. Names must not clash with the C reserved words

Where Variables are Declared

Outside the main program and functions

These are **global variables** as they can be used in the main function and any other function defined in the source file after the declaration.

At the start of main or other functions

These are called **local variables** and are declared following the opening { in the function. Note that the declarations must come before any other statements in the function except following a further { as given below. They can only be used in the function where they are declared.

Following any other { in a function

These variables can only be used before the corresponding }.

Both inside and outside main or other function

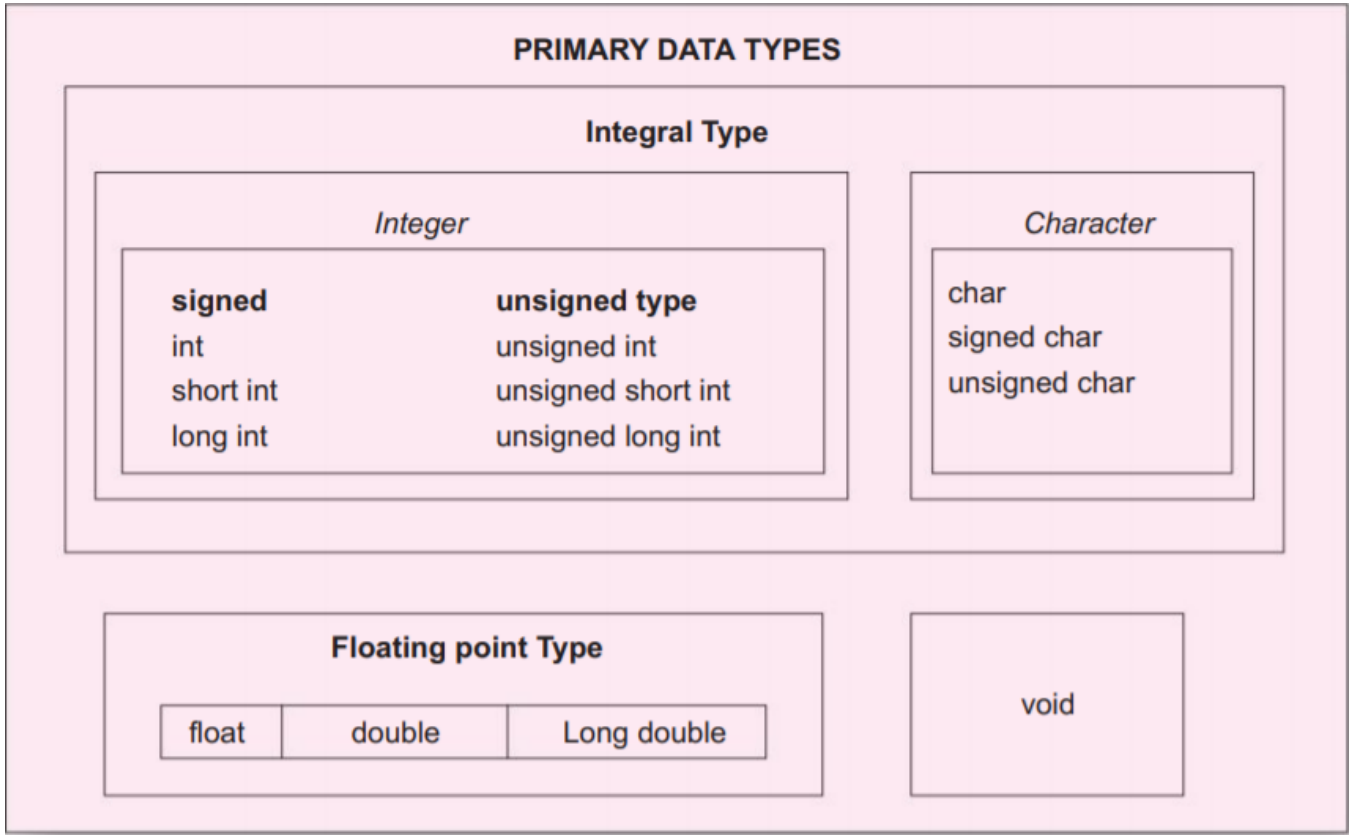
In this case two memory locations are reserved with the same name, but the local variable is always used in preference to the global variable where possible.

C data types

- The Data type is a set of value with predefined characteristics.
- Data types are used to declare variable, constants, arrays, pointers, and functions.
- In the C programming language, data types are classified as follows:

1. Primary data types/Predefined data types/Primitive
2. Derived data types (structure, array, pointer, function)
3. User-defined data types (typedef: keyword for defining datatypes)
4. Void data type

Primary Datatypes:



Datatype sizes:

Variable type	Number of bits
`char`	8
`int`	16 or 32 (usually)
`short int`	16 (usually)
`short`	16 (usually)
`long int`	32 (usually)
`long`	32 (usually)
`float`	about 32
`double`	about 64
`long float`	about 64
`long double`	> 64

Format specifiers in C

- Used for input and output operation.
- Way to make compiler know type of data is in a variable during taking input using the scanf() function and printing using printf() function.

Below mentioned elements lets us manage printing format:

- A minus(-) sign tells left alignment.
- A number after **%** specifies the minimum field width to be printed if the characters are less than the size of width the remaining space is filled with space and if it is greater than it printed as it is without truncation.
- A period(.) symbol separate field width with the precision.

Note: Precision tells the maximum number of digits in integer, characters in string and number of digits after decimal part in floating value.

Format Specifier	Type
%c	Character
%d	Signed integer
%e or %E	Scientific notation of floats
%f	Float values
%g or %G	Similar as %e or %E
%hi	Signed integer (short)
%hu	Unsigned Integer (short)
%i	Unsigned integer
%l or %ld or %li	Long
%lf	Double
%Lf	Long double
%lu	Unsigned int or unsigned long
%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character

Declaration of variables

- It lets a compiler know:

1. Name of a variable
2. Type of data to be stored (accordingly memory is allotted)

- Syntax:

<data-type> v1,v2,....vn ;

Separate variable names by comma (,) in case of declaring multiple variables in a statement.

Default values of constants

Integer constants, by default, represent **int** type data. We can override this default by specifying unsigned or long after the number (by appending U or L) as shown below:

Literal	Type	Value
+111	int	111
–222	int	–222
45678U	unsigned int	45,678
–56789L	long int	–56,789
987654UL	unsigned long int	9,87,654

Similarly, floating point constants, by default represent **double** type data. If we want the resulting data type to be **float** or **long double**, we must append the letter f or F to the number for **float** and letter l or L for **long double** as shown below:

Literal	Type	Value
0.	double	0.0
.0	double	0.0
12.0	double	12.0
1.234	double	1.234
–1.2f	float	–1.2
1.23456789L	long double	1.23456789