0301102 LOGIC DEVELOPMENT & PROGRAMMING

UNIT – 2

Data Types

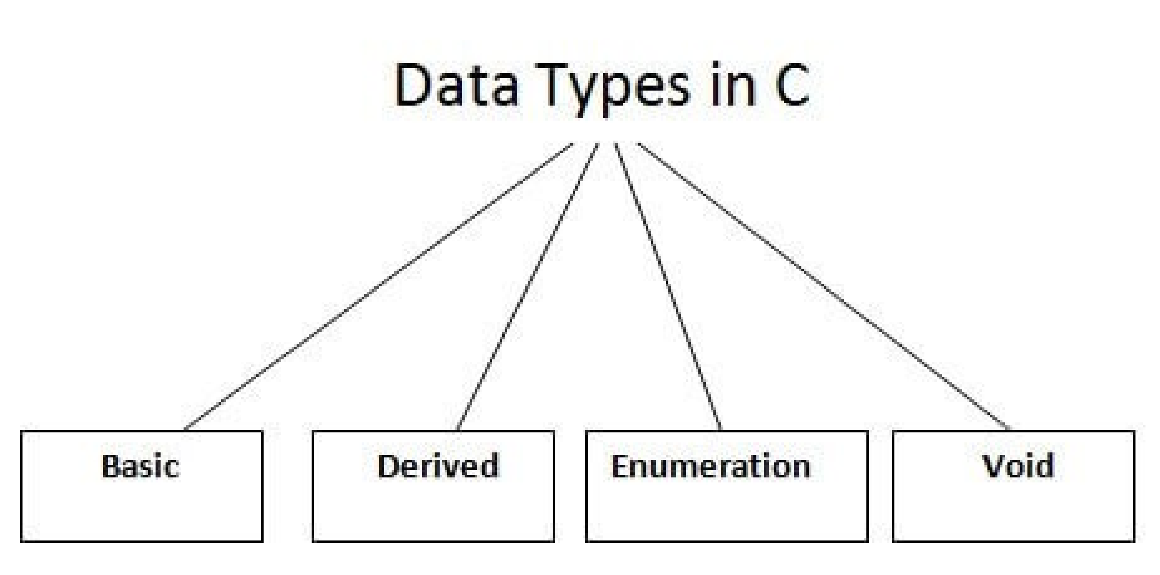
-

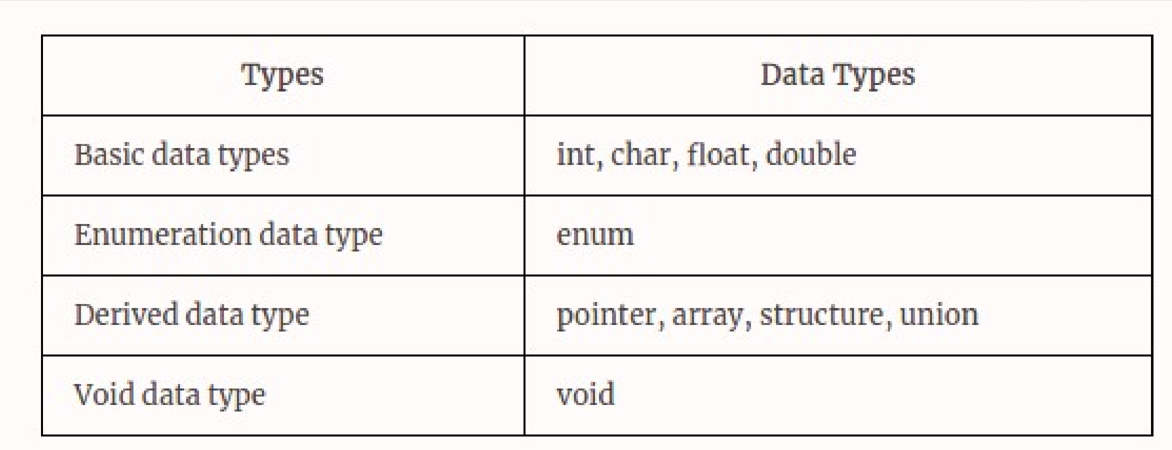
Compile and Execute C Program

- Open a text editor and add the c code.
- Save the file as hello.c (filename.c)
- Open a command prompt and go to the directory where you have saved the file.
- Type gcc/g++ hello.c and press enter to compile your code.
- If there are no errors in your code, the command prompt will take you to the next line and would generate a.out executable file.
- Now, type ./a.out to execute your program.
- You will see the output "Hello World" printed on the screen.

Data Types

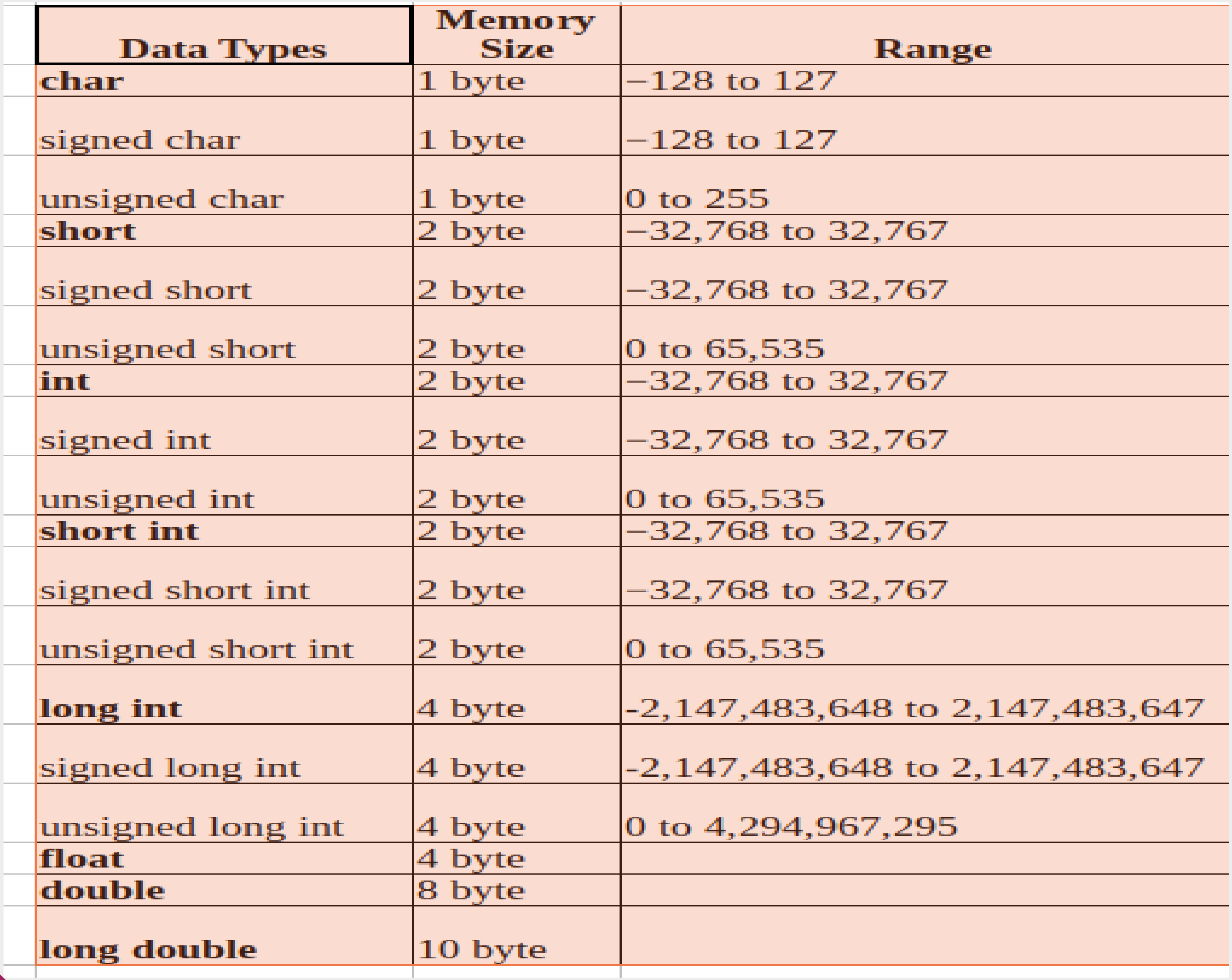
- A data type specifies the type of data that a variable can store such as integer, floating, character, etc.
- Data types are used to define a variable before to use in a program.
- There are four data types in C language. They are:





Types	Data Types
Basic data types	int, char, float, double
Enumeration data type	enum
Derived data type	pointer, array, structure, union
Void data type	void

Data Types



Data Types	Memory Size	Range
char	1 byte	–128 to 127
signed char	1 byte	–128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	–32,768 to 32,767
signed short	2 byte	–32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	–32,768 to 32,767
signed int	2 byte	–32,768 to 32,767
unsigned int	2 byte	0 to 65,535
short int	2 byte	–32,768 to 32,767
signed short int	2 byte	–32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

Data Types

1) **Basic data types in C language:**

- **1.1. Integer data type:**
- Integer data type allows a variable to store **numeric values.**
- “int” keyword is used to refer integer data type.
- The storage size of int data type is 2 or 4 or 8 byte.
- It varies depend upon the processor in the CPU that we use. If we are using 16 bit processor, 2 byte (16 bit) of memory will be allocated for int data type.
- Like wise, 4 byte (32 bit) of memory for 32 bit processor and 8 byte (64 bit) of memory for 64 bit processor is allocated for int datatype.
- int (2 byte) can store values from -32,768 to +32,767
- int (4 byte) can store values from -2,147,483,648 to +2,147,483,647.

Note:

- We can’t store decimal values using **int** data type.
- If we use int data type to store decimal values, decimal values will be truncated and we will get only whole number.
- In this case, float data type can be used to store decimal values in a variable.

Data Types

1) **Basic data types in C language:**

   - **1.2. Character data type:**
   - Character data type allows a variable to store **only one character.**
   - Storage size of character data type is 1 byte. We can store only one character using character data type.
   - “char” keyword is used to refer character data type.
   - For example, ‘A’ can be stored using char datatype. You can’t store more than one character using char data type.

Data Types

1) **Basic data types in C language:**

- **1.3. Floating point data type:**
- Floating point data type consists of 2 types. They are
- float
- double

1. float

- Float data type allows a variable to store **decimal value**s.
- Storage size of float data type is 4 byte. This also varies depend upon the processor in the CPU as “int” data type.
- We can use up-to 6 digits after decimal using float data type.
- For example, 10.456789 can be stored in a variable using float data type.

2. double:

- Double data type is also same as float data type which allows up-to 10 digits after decimal.
- The range for double datatype is from 1E–37 to 1E+37.

Data Types

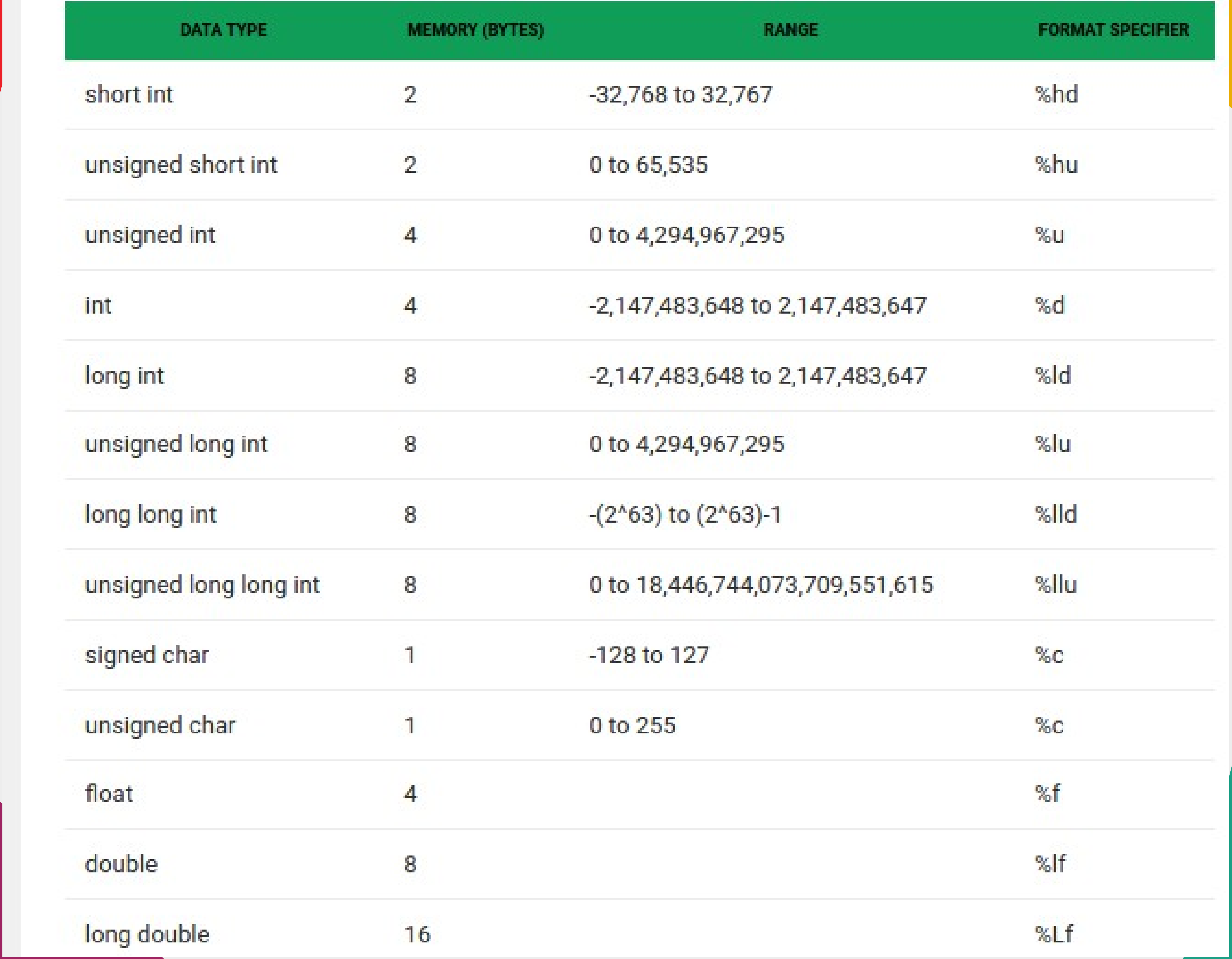
Signed and Unsigned Data Types

- **Signed variables,** such as signed integers will allow you to represent numbers both in the positive and negative ranges.
- **Unsigned variables**, such as unsigned integers, will only allow you to represent **numbers in the positive.**
- Unsigned and signed variables of the same type (such as int and char) both have the same range (range of 65,536 and 256 numbers, respectively), but unsigned can represent a larger magnitude number than the corresponding signed variable.
- For example, an **unsigned char** can represent values from 0 to 255, while **signed char** can represent -128 to 127.

Format Specifier:

- **%d is for int**
- **%c is for char**
- **%f is for float**

Data Types – Format Specifier



DATA TYPE	MEMORY (BYTES)	RANGE	FORMAT SPECIFIER
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	8	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	8	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	16		%Lf

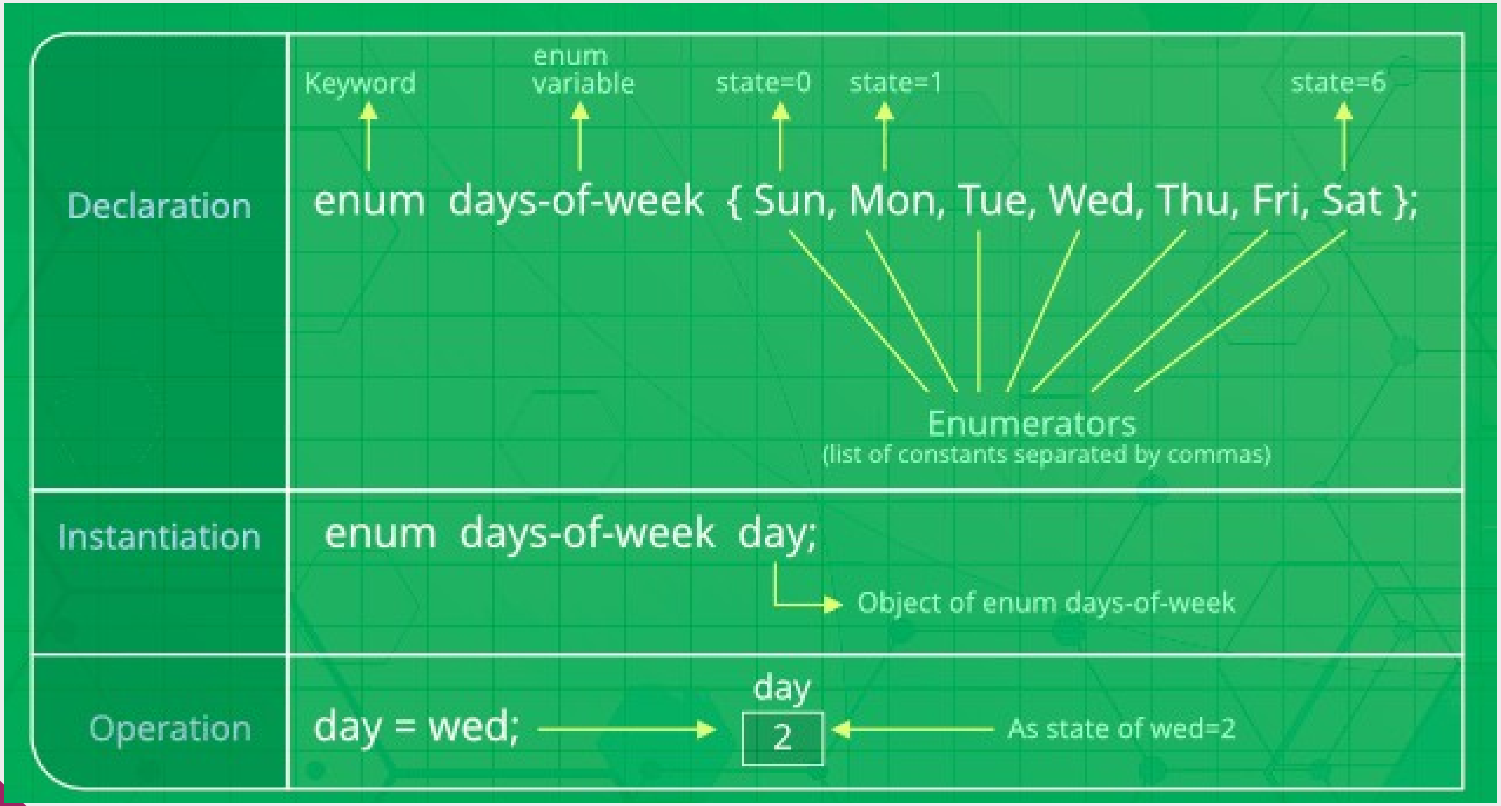
Data Types

- **sizeof() function in C language:**
- sizeof() function is used to find the memory space allocated for each C data types.
- **Example:**

```
#include <stdio.h>
int main()
{
   int a;
   char b;
   float c;
   double d;
   printf("Storage size for int data type:%d \n",sizeof(a));
   printf("Storage size for char data type:%d \n",sizeof(b));
   printf("Storage size for float data type:%d \n",sizeof(c));
   printf("Storage size for double data type:%d\n",sizeof(d));
   return 0;
}
```

Data Types

Enumeration data type

- Enumeration data type consists of named integer constants as a list.
- It start with 0 (zero) by default and value is incremented by 1 for the sequential identifiers in the list.
- **Enum syntax in C:**
  - enum var_name {const1, const2, ..., constN};
- 

Data Types

Enum EXAMPLE:

```
#include<stdio.h>
enum week{Mon=1, Tue, Wed, Thur, Fri, Sat, Sun};
int main()
{
    enum week today; //creating today variable of enum week type
    today = Wed;
    printf("%d",today);
}
```

Data Types

Derived data type in C language:

- Array, pointer, structure and union are called derived data type in C language.

Void data type in C language:

- Void is an empty data type that has no value.
- This can be used in functions and pointers.

Unformatted Input Output

Getchar():

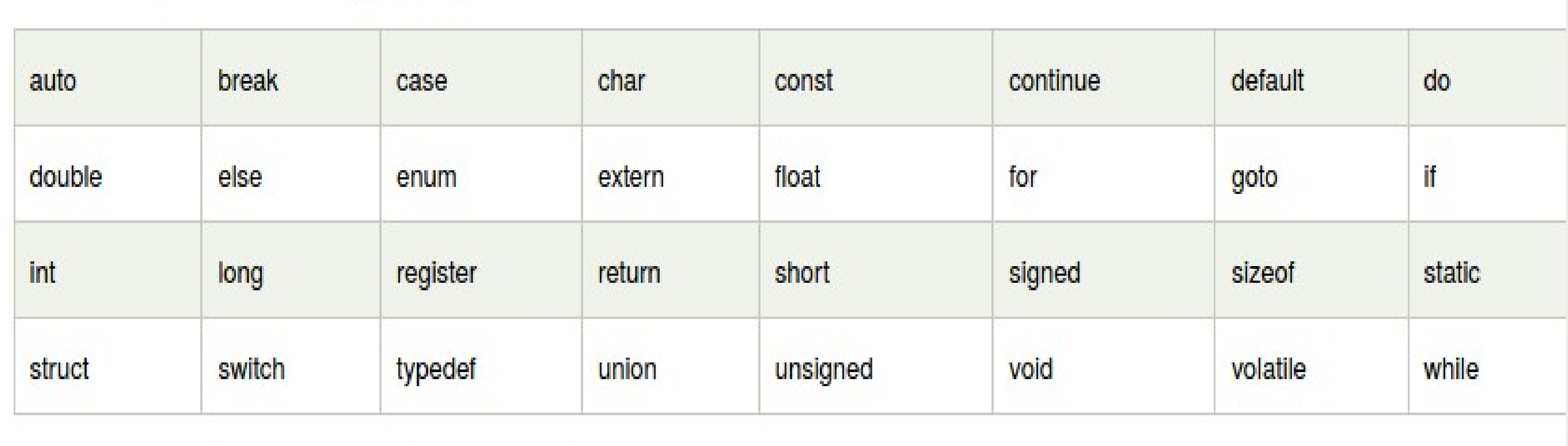
- A getchar() function is a non-standard function whose meaning is already defined in the stdin.h header file to accept a single character input from the user.
- A getchar() reads a single character from standard input.
- Syntax: int getchar ();
  - it returns the read characters as an unsigned char in an int, and if there is an error on a file, it returns the EOF at the end of the file.

Putchar()

- The putchar() function displays the character passed to it on the screen and returns the same character. This function too displays only a single character at a time.
- Syntax: int putchar(int char)
- Parameters: This method accepts a mandatory parameter char which is the character to be written to stdout.

Keywords in C

- A keyword is a **reserved word**. You cannot use it as a variable name, constant name, etc. **There are only 32 reserved words (keywords) in the C language**.
- A list of 32 keywords in the c language is given below:



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

Input/Output Functions - printf() and scanf() in C

- The printf() and scanf() functions are used for input and output in C language. Both functions are inbuilt library functions, defined in stdio.h (header file).
- **printf() function:** The printf() function is used for output. It prints the given statement to the terminal.
- The syntax of printf() function is given below:
  - **printf("format string",argument_list);**
  - The format string can be %d (integer), %c (character), %s (string), %f (float) etc.
- **scanf() function:** The scanf() function is used for input. It reads the input data from the terminal.
- The syntax of scanf() function is given below:
  - scanf("format string",argument_list);

printf() and scanf() in C

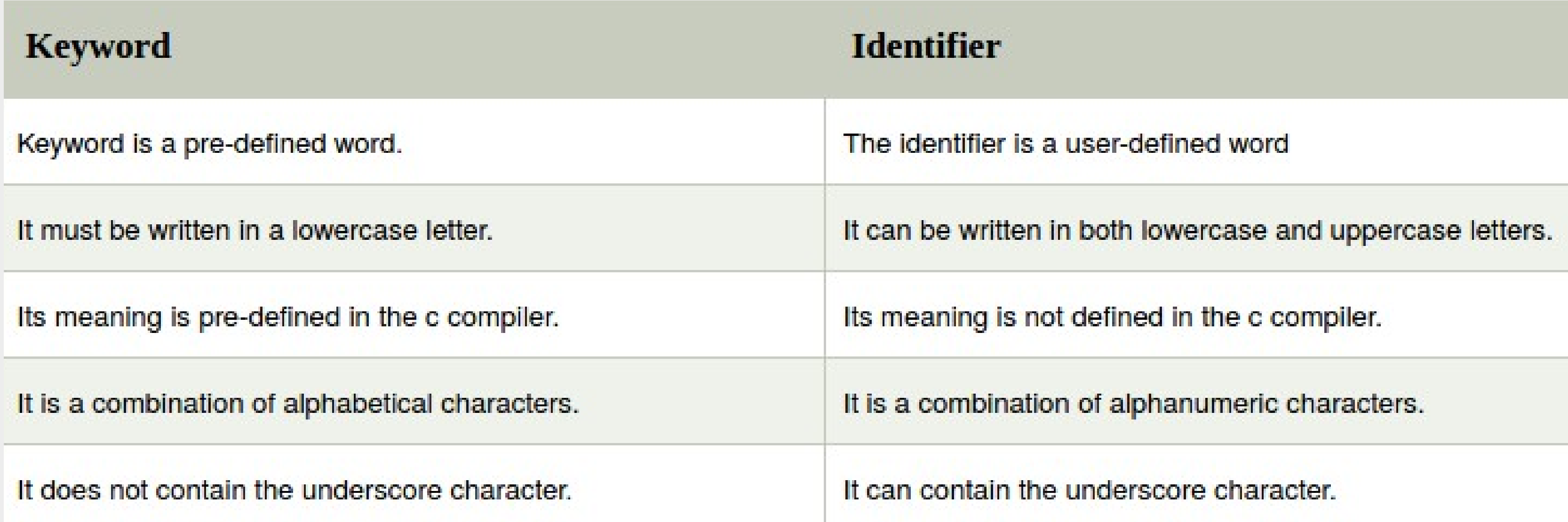
```
#include<stdio.h>
int main()
{
   int x=0,y=0,result=0;
   printf("enter first number:");
   scanf("%d",&x);
   printf("enter second number:");
   scanf("%d",&y);
   result=x+y;
   printf("sum of 2 numbers:%d ",result);
}
```

C Identifiers

- C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc.
- An identifier can be composed of letters such as uppercase, lowercase letters, underscore, digits, but the starting letter should be either an alphabet or an underscore.
- **Rules for constructing C identifiers**
- The first character of an identifier should be either an **alphabet** or an **underscore**, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any **numerical digit**.
- Identifiers are **case sensitive.**
- **Commas** or **blank spaces** cannot be specified within an identifier.
- **Keywords** cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers name should be such that it is meaningful, short, & easy to read.

C Identifiers

- **Types of identifiers**
  - Internal identifier
  - External identifier
- If the identifier is not used in the external linkage, then it is known as an internal identifier. The **internal identifiers** can be local variables.
- If the identifier is used in the external linkage, then it is known as an external identifier. The **external identifiers** can be function names, global variables.



Keyword	**Identifier**
Keyword is a pre-defined word.	The identifier is a user-defined word
It must be written in a lowercase letter.	It can be written in both lowercase and uppercase letters.
Its meaning is pre-defined in the c compiler.	Its meaning is not defined in the c compiler.
It is a combination of alphabetical characters.	It is a combination of alphanumeric characters.
It does not contain the underscore character.	It can contain the underscore character.

ASCII

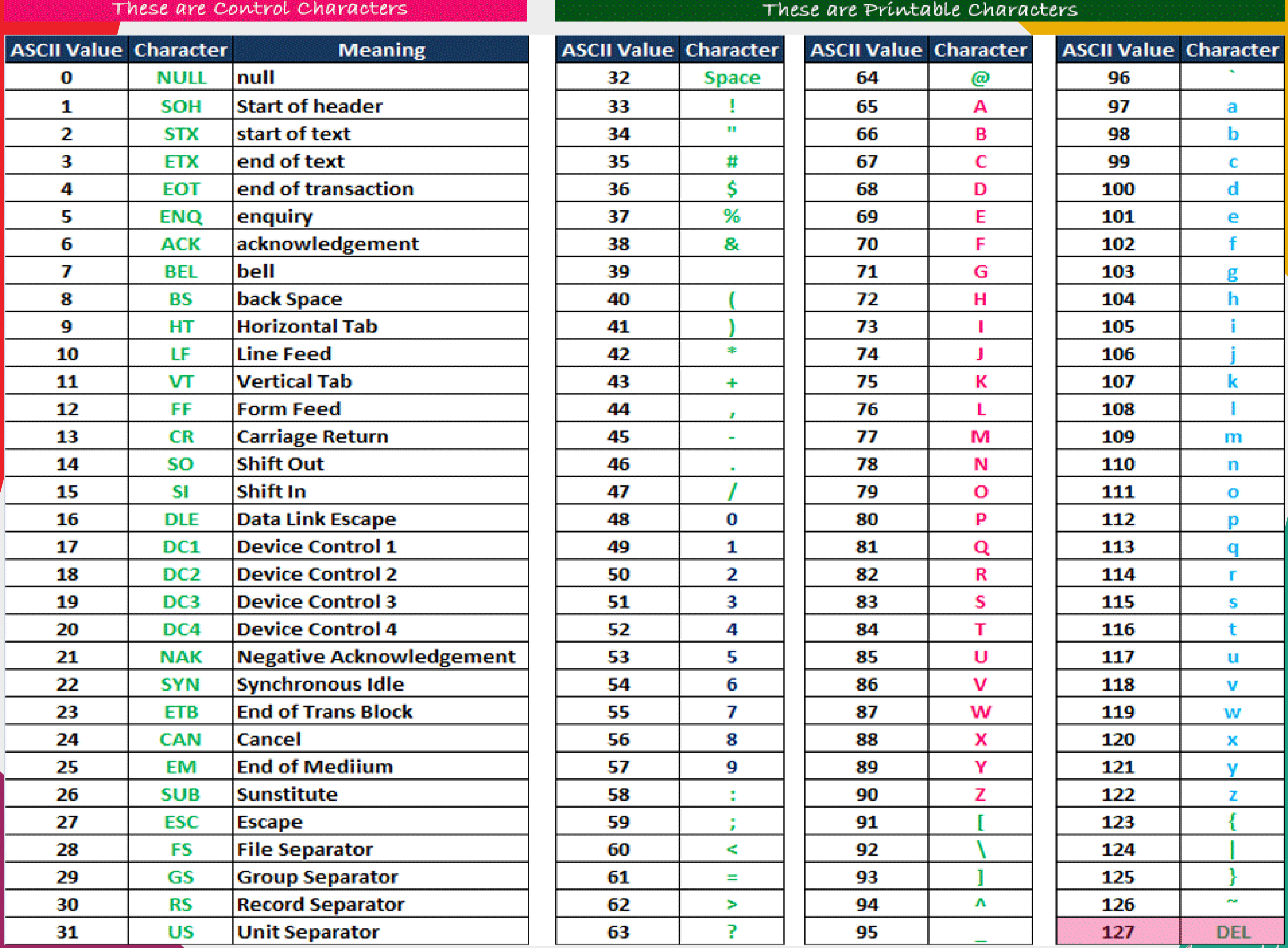
- The full form of ASCII is the **American Standard Code for information interchange.**
- As we are humans we have our language to understand the same way machine also have the same thing to understand **characters, digits, special characters** that is ASCII representation of the character.
- It is a character encoding scheme used for electronics communication. Each character or a special character is represented by some ASCII code, and each ascii code occupies 7 bits in memory.
- These characters include **upper and lowercase English letters, numbers, and punctuation symbols.**
- **Standard ASCII can represent 128 characters**.
- The ascii value represents the character variable in numbers, and each character variable is assigned with some number range from 0 to 127. For example, the ascii value of 'A' is 65.
- ASCII contains numbers, each character has its own number to represent. We have 256 character to represent in C (0 to 255) like character (a-z, A-Z), digits (0-9) and special character like !, @, # etc.

Character Set

- C language has a set of characters which include alphabets, digits, and special symbols. C language 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.
  - Alphabets
  - Digits
  - Special Symbols
- **Alphabets**
  - C language supports all the alphabets from the English language. Lower and upper case letters together support 52 alphabets.
  - lower case letters - a to z
  - UPPER CASE LETTERS - A to Z

Character Set

- **Digits**
  - C language supports 10 digits which are used to construct numerical values in C language.
  - Digits - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- **Special Symbols**
  - C language supports a rich set of special symbols that include symbols to perform mathematical operations, to check conditions, white spaces, backspaces, and other special symbols.
  - Special Symbols - ~ @ # $ % ^ & * ( ) _ - + = { } [ ] ; : ' " / ? . > , < \ | tab newline space NULL bell backspace verticaltab etc.,



These are Control Characters

ASCII Value	Character	Meaning
0	NULL	null
1	SOH	Start of header
2	STX	start of text
3	ETX	end of text
4	EOT	end of transaction
5	ENQ	enquiry
6	ACK	acknowledgement
7	BEL	bell
8	BS	back Space
9	HT	Horizontal Tab
10	LF	Line Feed
11	VT	Vertical Tab
12	FF	Form Feed
13	CR	Carriage Return
14	SO	Shift Out
15	SI	Shift In
16	DLE	Data Link Escape
17	DC1	Device Control 1
18	DC2	Device Control 2
19	DC3	Device Control 3
20	DC4	Device Control 4
21	NAK	Negative Acknowledgement
22	SYN	Synchronous Idle
23	ETB	End of Trans Block
24	CAN	Cancel
25	EM	End of Mediium
26	SUB	Sunstitute
27	ESC	Escape
28	FS	File Separator
29	GS	Group Separator
30	RS	Record Separator
31	US	Unit Separator

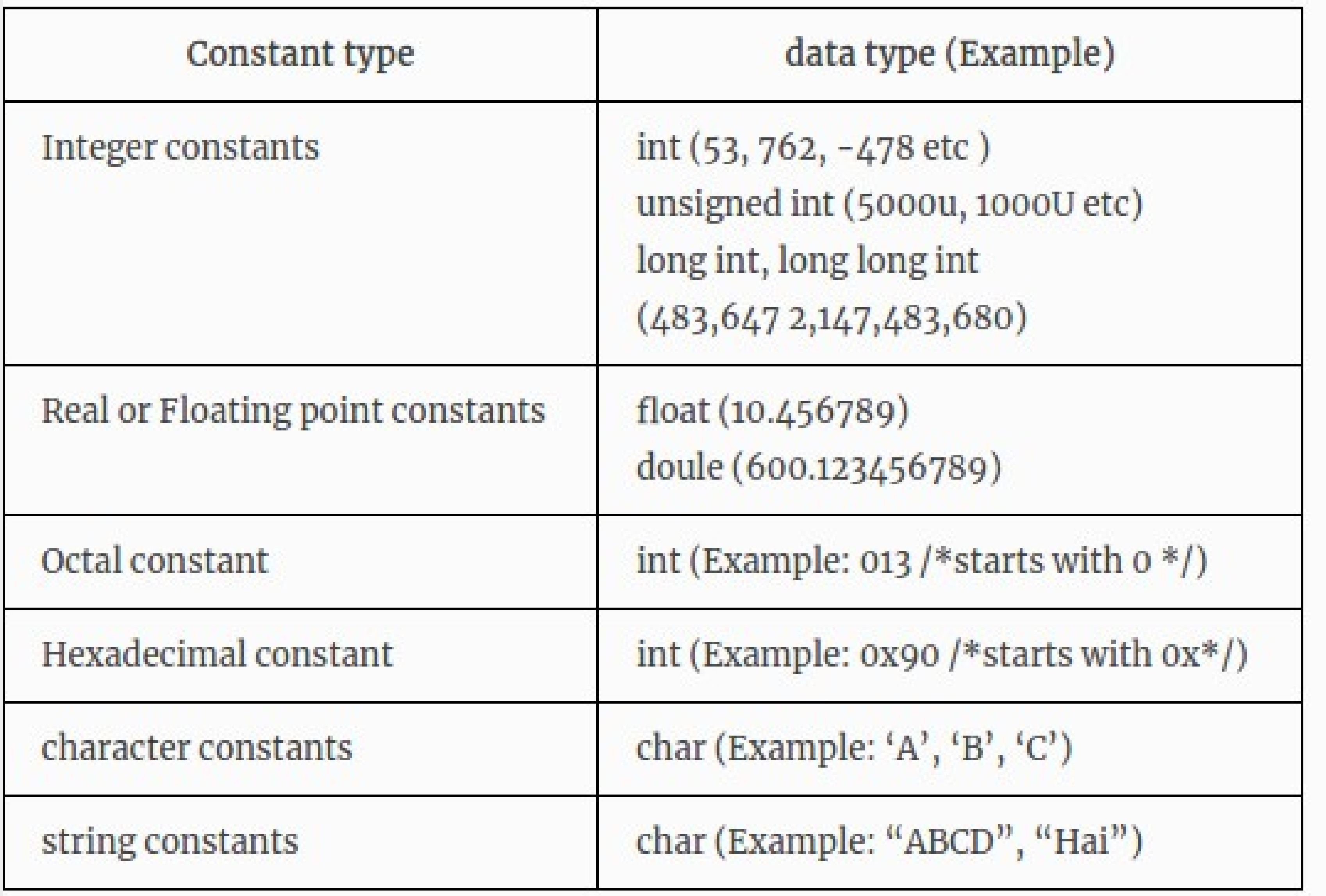
These are Printable Characters

ASCII Value	Character	ASCII Value	Character	ASCII Value	Character
32	Space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39		71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

Constants

- C Constants are also like normal variables. But, only difference is, their values can not be modified by the program once they are defined.
- Constants refer to fixed values. They are also called as **literals.**
- Constants may be belonging to any of the data type.
- **Syntax:**
  - const data_type variable_name;
- **Types of C constant:**
  - Integer constants
  - Real or Floating point constants
  - Octal & Hexadecimal constants
  - Character constants
  - String constants
  - Backslash character constants

Constants



Constant type	data type (Example)
Integer constants	int (53, 762, -478 etc) unsigned int (5000u, 1000U etc) long int, long long int (483,647 2,147,483,680)
Real or Floating point constants	float (10.456789) doule (600.123456789)
Octal constant	int (Example: 013 /*starts with 0 */)
Hexadecimal constant	int (Example: 0x90 /*starts with 0x*/)
character constants	char (Example: ‘A’, ‘B’, ‘C’)
string constants	char (Example: “ABCD”, “Hai”)

Constants

- Rules for constructing C constant:

1. Integer Constants in C:

- An integer constant must have at least one digit.
- It must not have a decimal point.
- It can either be positive or negative.
- No commas or blanks are allowed within an integer constant.
- If no sign precedes an integer constant, it is assumed to be positive.
- The allowable range for integer constants is -32768 to 32767.

2. Real constants in C:

- A real constant must have at least one digit
- It must have a decimal point
- It could be either positive or negative
- If no sign precedes an integer constant, it is assumed to be positive.
- No commas or blanks are allowed within a real constant.

Constants

3. Character and string constants in C:

- A character constant is a single alphabet, a single digit or a single special symbol enclosed within single quotes.
- The maximum length of a character constant is 1 character.
- String constants are enclosed within double quotes.

4. Backslash Character Constants in C:

- There are some characters which have special meaning in C language.
- They should be preceded by backslash symbol to make use of special function of them.
- Like - \b – Backspace, \n - New line, \t - Horizontal tab, etc...

Constants

How to use constants in a C program?

- We can define constants in a C program in the following ways.
  1) **By “const” keyword**
  2) **By “#define” preprocessor directive**
- **Note:** when you try to change constant values after defining in C program, it will through error.

EXAMPLE:

```
#include<stdio.h>

int main()

{

    const float PI=3.14;

    printf("The value of PI is: %f",PI);

    return 0;

 }
```

Constants

EXMAPLE:

```
   #include <stdio.h>
   #define PI 3.14
  int main()
 {
     printf("%f",PI);
 }
```