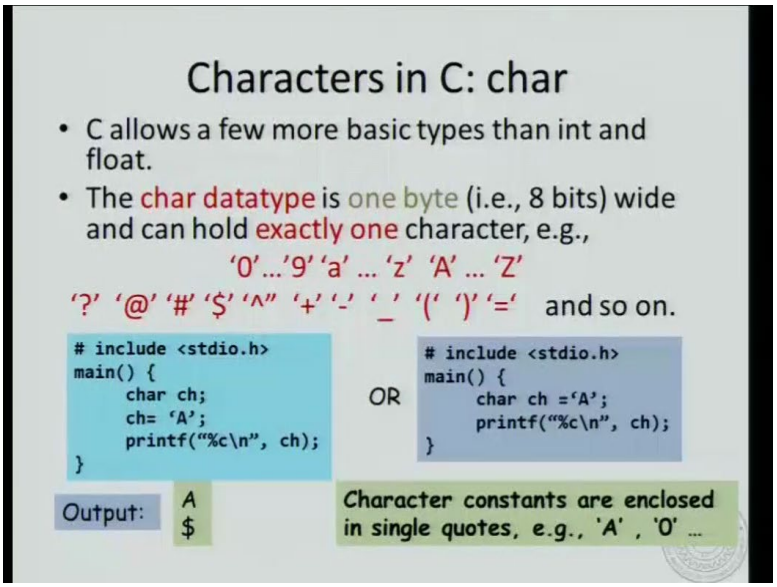
Introduction to Programming in C
Department of Computer Science and Engineering

In this session, we will learn about one more fundamental data type in C. So, far we have seen ints and floats. Ints are supposed to represent integers and floats are supposed to represent real numbers. We will see the third most important data type which is character. So, it is called char in c or char.

(Refer Slide Time: 00:23)

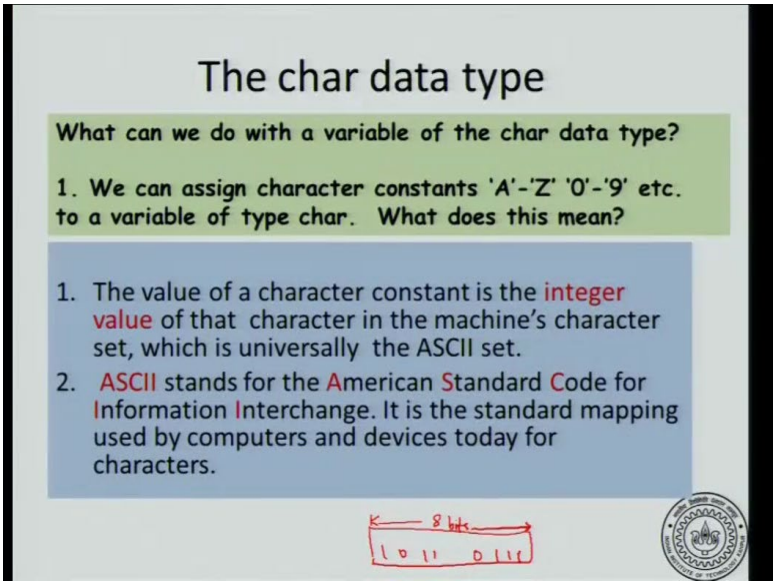


C allows a character data type to be 1 byte that is 8 bits wide, and 1 byte can hold exactly one character. For example, a character may be a digit like 0 so, on up to 9. It can be lower case letter like a up to z, it can be upper case letter like capital A through capital Z and so, on. Similarly, there are other characters question marks and sharp and so, on. So, how do you declare a character variable, how do you assign it and how do you print or scan it. So, these are the basic operations that you can do with any data type. So, you declare a character variable using the data type char ch will declare variable of name ch and of data type char. In order to assign it to any particular constant, any particular character, what you have to do is, you write ch = 'A'. So, this is how you would assign any character in constants. All the character in constants are supposed to be enclosed in this single code. For example, '0' stands for the character 0 and not the number 0 and similarly, a within single code stands for character a.

Now, how do you prints print a characters you can use the format specifier %c. So, recall that %d prints an integer and %f prints of float, we have the third fundamental data type

which is character which can be printed using a %c. So, if you say print f %c ch, it will print a. There is also an abbreviator notation where as soon as you declare the variable, you can initialize it using character ch equal to a. This is similar to saying int I equal to zero. It is the same concept.

(Refer Slide Time: 02:49)

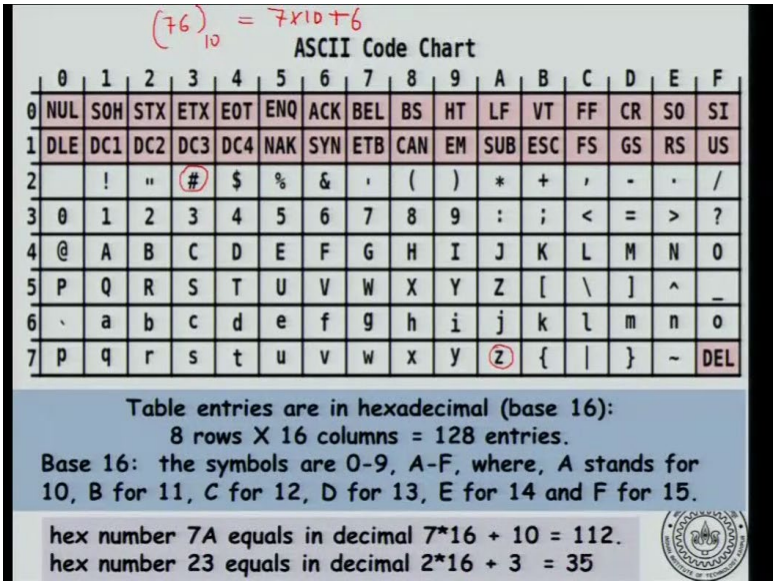


Now, what can we do with a character data type? For example, we can assign character constants to those characters variables. Now, what does a character variable mean? Here is the first surprise. The value of a character constant is an integer that the machines represent, machine stores which is usually the ASCII set. What does this mean? The machine deals with fundamentally bits. So, you have a data field which is 8 bits wide and this is sequence of bits say 1 0 1 1 0 1 1 1.

Now, here is the bit pattern and if you see that this bit pattern is a char, then the machine takes this integer, takes this bit pattern as an integer and looks up a table known as the ASCII set table and sees which character it is. So, the value of the character constant is actually an integer. What does that integer represents? The integer represents a particular entry in an ASCII character table and what entry is in that particular location, that is the character constant. So, think of it like the following. The character is just an uninterpreted sequence of bites. If you tell the machine, please read this as an integer, it will read this as an integer. If you read this, if you tell the machine please read this as a character, it will take that integer, go look up the ASCII table and see that this integer stands for the character c and prints that. So, by itself the bit pattern can be interpreted in multiple ways.

So, here is a surprising thing which is different from natural language. There are certain natural languages where this does not typically happen with Indian languages, but there are certain languages where you have a character and how you read it depends on where you saw it. So, if it was in the middle of a text, then this is an alphabet. If you saw this in the middle of a numbers sequence, then it is a number. What happens in the machine is somewhat similar. You have a bit sequence and this thing is interpreted as a character by looking up the ASCII set. ASCII stands for American Standard Code for Information Interchange, and it is one of the popular encodings for characters used in computers. So, the code chart looks something like this.

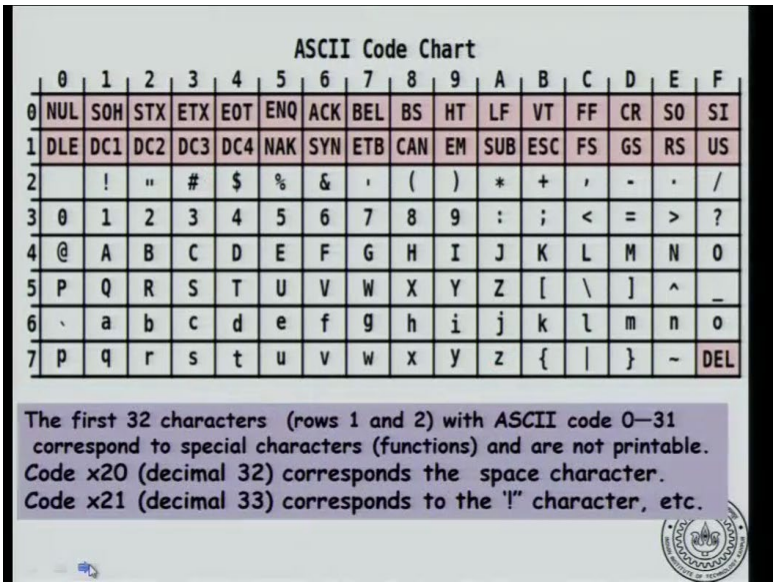
(Refer Slide Time: 05:56)



You have 256 characters and characters can be looked up in a table. The table entries are in hexadecimal so, base 16. We will come to that little in the course why basic 16 is convenient,, but there are 8 rows and 16 columns in the table. So, in base 16 notation, a stands for 10, b stands for 11, c for 12 so, on up to f for 15. So, this is what is meant by base 16 notation.

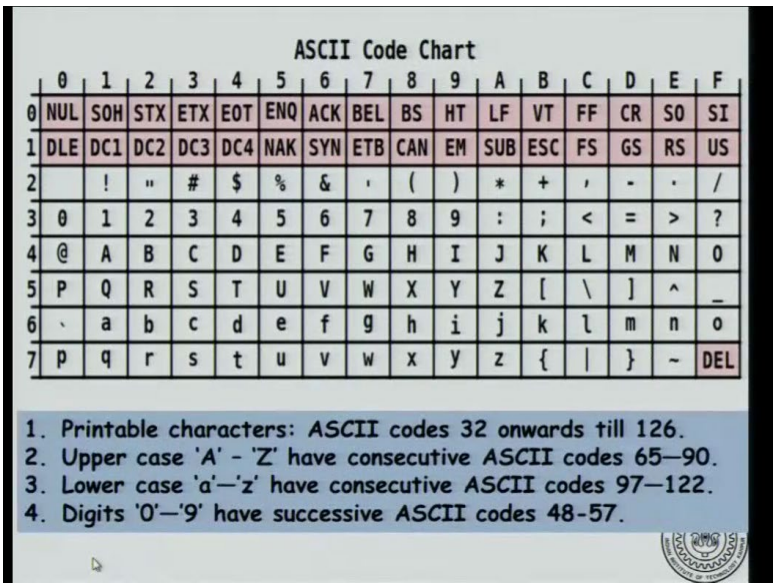
So, let us look at what does the number 7 a represent. 7 a is row 7 column number 10. So, that is the number that I am interested in. What does 7 a represents? It means 7 times 16 plus 10. So, in base 10 notations, the number 76 let us say so, if I have this number 7 in base 10 notation, this; obviously, stands for the numerical values 7 into 10 plus 6. Similarly, in base 16 notation, 7 a stand for 7 into 16 plus 10. Remember that a is 10. So, you have 112 and similarly, hexadecimal 2 3. So, row 2 column 3 for example, hexadecimal 2 3 means look up 2 time 16 plus 3, the 35th entry in the table.

(Refer Slide Time: 07:51)



Now, here is the structure of the ASCII code set that you use in c, the first 32 characters basically from 0 0 hexadecimal to 1 f hexadecimal. So, these 32 characters which are shaded, are what are known as special characters, and they are not printable. They are required by the computer for certain special purposes. Code 2 0 that is decimal 32, 2 0 is 2 times 16 plus 0. So, this particular entry corresponds to the space characters. So, this is just a blank space. Code 21 corresponds to the exclamation character and so, on.

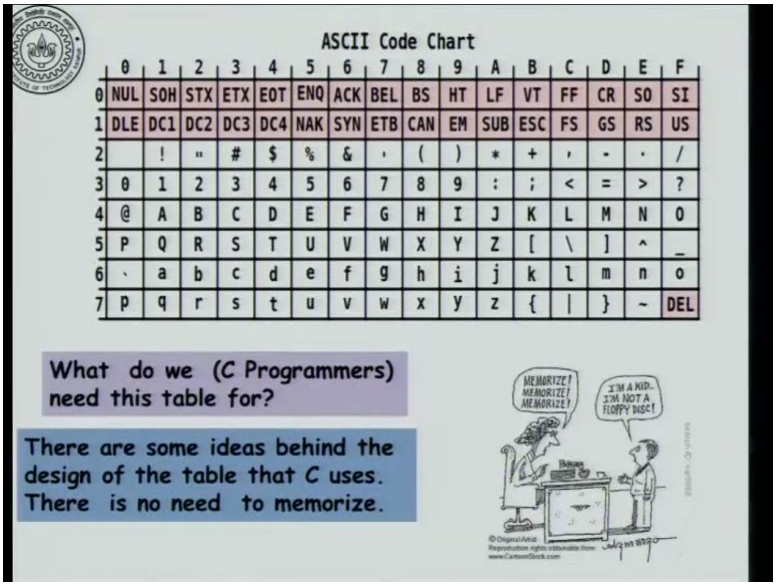
(Refer Slide Time: 08:40)



So, the printable characters in the ASCII code are hexadecimal 20, that is decimal 32 until 126. So, what is enclosed in the green parenthesis, these are all printable characters. Now, out of this, the capital letters start from x 41 which is 65 in decimal and go on up

till decimal 90. Small letters start from 97 and go on till 122 and so, on digits 0 to 9 occur before any character. So, why we need this information? This is how the characters are stored in the computer and do we really need to know it?

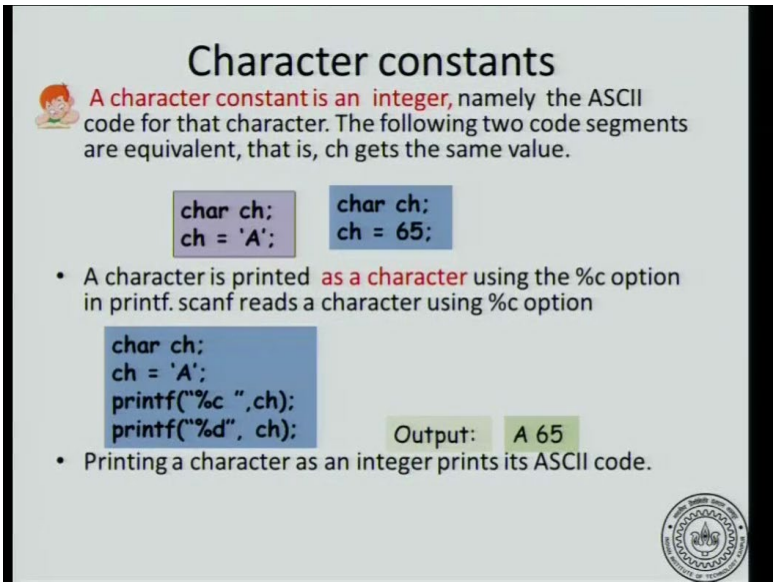
(Refer Slide Time: 09:46)



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

The point is not that you have to memorize this table. You do not need to memorize the table, but you need to remember certain abstract properties of the table. We will make that precise in a moment. We do not have to say that the ASCII code for a is 65 or 42 that is a waste of our memory. So, let us just see what we can do with this table without really remembering what that table looks like. So, there are some ideas behind the design of the table, how the table is structured which c programmers can use. There is no need to remember that a particular character has a particular ASCII value.

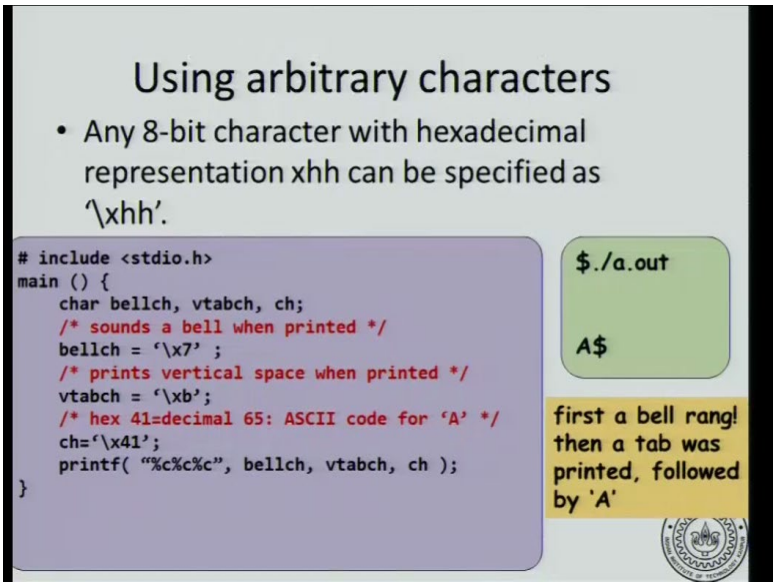
(Refer Slide Time: 10:37)



So, let us just recall. A character constant is an integer, namely the ASCII code for that character now which means that I will emphasize this with a very strange code. I can declare character ch and say char ch =‘A’ that; obviously, initializes the character to a. It assigns the value a to the variable ch, but I could also do the following characters ch = 65. Why 65? The ASCII value for a was 65. So, instead of writing it as a within single code, I can write ch = 65, and it will be the correct ASCII character anyway. Now, this means that the same character can also be interpreted as an integer if you really want to think of it that way.

So, for example, I can say %f %c ch if I do it in print f, it will print it as. So, the first print f will print a, but I could also take a character variable and ask c to print it as an integer using %d, it will print 65. So, remember that the external form that we see in some sense is the letter a. The internal representation is the number 65 because 65 is the entry in the ASCII table corresponding to the character a.

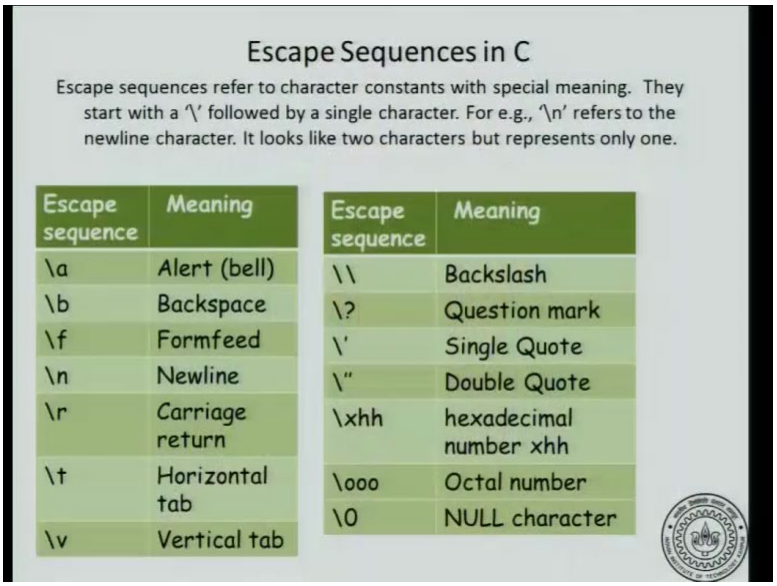
(Refer Slide Time: 12:22)



Now, one more thing is that you can print arbitrary numbers, even non-printable characters you can sort of print them using c and one way to do that is I can print any 8 bit character with a hexadecimal representation like \s, \x followed by the hexadecimal to digit. For example, \x followed by 7 is the bell character. So, let me go back a couple of times, couple of slides. So, if you look at the 7th entry in the ASCII table, it is represented as bell. It is a small bell in your system. So, if you ask the system to print the 7th character in the ASCII table, what will happen is that your computer will make a small beep sound. So, there are certain non-printable characters which can also be printed directly using… ok.

Similarly, let say \xb is the 11th number in the ASCII table, it is a vertical space. So, if you print that character, it prints a vertical space. Similarly, if I ask it to print hexadecimal 41 using \x41 so, x 41 is 4*16 + 1 which is 64 + 1=65 and we just saw that ASCII value 65 was the character a. So, if I ask it to print ch which is hexadecimal 41 as a character, then it will print the value a. So, when you run this program, what it will do is, first because you ask it to print a bell character, it will beep once, it will ring the bell and then, it will print the second character which is a vertical space. So, it will print a vertical space and then, the third character was a printable character a, it will print a. So, you can ask the system to print arbitrary entries in the ASCII table. If it is a printable character, it will print that corresponding character. If it is non-printable character, it might take a suitable action.

(Refer Slide Time: 15:07)



So, just for information sake, instead of printing it as \x followed by the x code, c provides certain escape characters, some special sequences as well in order to print these non-printable characters. First of all until now we have seen one such number which is \n. So, \n is the new line character. It is a non-printable character, but it corresponds to some ASCII corrected. Similarly, for the other non-printable characters, c has some escape characters. For example, back slash a is the bell character and so, on.