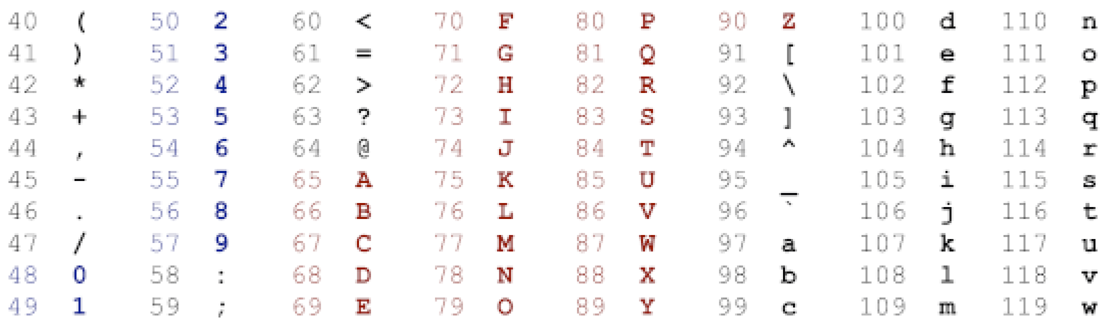
char

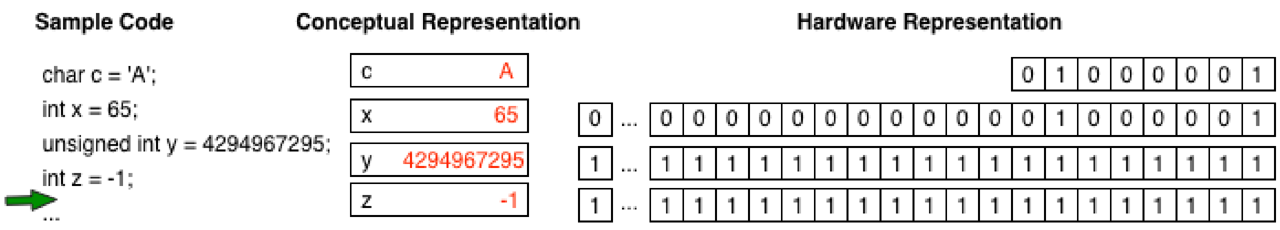
coursera.org/learn/programming-fundamentals/supplement/Xmf4T/char

A char (pronounced either “car” or “char”) is the smallest data type—a mere 8 bits long—and is used to encode characters. With only 8 bits, there are only 2^{8}=256 possible values for a char (from 00000000 to 11111111). On most machines you will use, these 8 bits are interpreted via the American Standard Code for Information Interchange (or ASCII) character-encoding scheme, which maps 128 number sequences to letters, basic punctuation, and upper- and lower-case letters. A subset of this mapping is shown in the figure below - please don’t try to memorize it. Another, much more expressive character-encoding scheme you may encounter (particularly when needing to encode non-English characters) is Unicode (which requires more than 1 byte).



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

Now if you look at the first line of code in this next figure:



```
char c = 'A';
int x = 65;
unsigned int y = 4294967295;
int z = -1;
...
```

you can see the char c declared and initialized to the value 'A'. Notice that we wrote A in single quotation marks—these indicate a character literal. In the same way that we could write down a string literal previously, we can also write down a character literal: the specific constant character value we want to use. Writing down this literal gives us the numerical value for A without us having to know that it is 65. If we did need to know, we could consult an ASCII table like the one in the figure above. Being able to write 'A' instead of 65 is another example of abstraction—we do not need to know the ASCII encoding, we can just write down the character we want.

Figure caption: Examples of chars and ints. At first glance, c and x appear identical since they both have the binary value 65. However, they differ in both size (c has only 8 bits whereas x has 32) and interpretation (c’s value is interpreted using ASCII encoding whereas x’s value is interpreted as an integer). Similarly, y and z are identical in hardware but have differing interpretations because y is unsigned and z is not.