Bytes

Bits are almost always bundled into 8-bit Collections called **Bytes**.

One byte can be represented by 2 Hex digits

A	F	
1010	1111	– 1 byte (2 nibbles)

With 8 bits, we can represent $2^8 = 256$ different 'things'

for example we can represent the numbers 0 to 255

A single byte is also sufficient to represent all the characters in a western keyboard.

The American System for Coded Information Interchange (ASCII) Character Set is a standardized table in which each number from 0 to 255 is associated with either a **symbol** (that we call a **printable character**) or with an **action** (that we call an **unprintable character**)

Ascii character set.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Extended ASCII Codes

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	|	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	—	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	·
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	▒	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

This table is effectively a way to associate a character with a number. Printable characters mean something to us, as humans, but apart from that, there is nothing special about them.

Note the distinction between the character '5' and the number 5.

The 8 bit representation for the character '5' is:

00110101 (i.e., 35_{hex})

Whereas the number 5 is:

00000101 (i.e., 05_{hex})

Representing Characters in a Program.

Consider the following C Program fragment in which we declare 2 variables: ch, of type character, and i, of type integer.

The Program also Contains a number of Constants:
'5', 53, 0x35, 065.

```
char ch;
int i;

int main(){

    ch ='5';
    i = '5';
    printf("ch as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", ch, ch, ch, ch);
    printf("i  as an int = %d, in hex = %x,in octal = %o and as a char = %c\n", i, i, i, i);
    printf("-------------------------------------------------------------\n");

    ch = 53;
    i = 53;
    printf("ch as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", ch, ch, ch, ch);
    printf("i  as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", i, i, i, i);
    printf("-------------------------------------------------------------\n");

    ch = 0x35;
    i = 0x35;
    printf("ch as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", ch, ch, ch, ch);
    printf("i  as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", i, i, i, i);
        printf("-------------------------------------------------------------\n");

    ch = 065;
    i = 065;
    printf("ch as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", ch, ch, ch, ch);
    printf("i  as an int = %d, in hex = %x, in octal = %o and as a char = %c\n", i, i, i, i);
    printf("-------------------------------------------------------------\n");
```

'5' is a character Constant

53 is a decimal number Constant

0x35 is a hex number Constant

065 is an Octal number Constant

All of these Constants have the same bit Sequence: 00110101

In other words, they all have the same "Count value" – the distance from 0 to each number, in its respective base, is the same.

Our program assigns these Constants, in turn, to both the character Variable and to the Integer Variable and it Prints the values of those variables in decimal, hex, octal and character representations.

The Program output for this fragment is:

ch = '5' i = '5'	ch as an int = 53, in hex = 35, in octal = 65 and as a char = 5 i as an int = 53, in hex = 35,in octal = 65 and as a char = 5
ch = 53 i = 53	ch as an int = 53, in hex = 35, in octal = 65 and as a char = 5 i as an int = 53, in hex = 35, in octal = 65 and as a char = 5
ch = 0x35 i = 0x35	ch as an int = 53, in hex = 35, in octal = 65 and as a char = 5 i as an int = 53, in hex = 35, in octal = 65 and as a char = 5
ch = 065 i = 065	ch as an int = 53, in hex = 35, in octal = 65 and as a char = 5 i as an int = 53, in hex = 35, in octal = 65 and as a char = 5

Regardless of the representation we use, each variable is given the same bit sequence and, when printing, we can write that sequence out in any one of the desired representations.

Does this mean that Integer variables and character variables are the same?

?

At least at the bit representation level, they appear to be.

There are, however, some important differences

(1) Character variables are only 1-byte in size

Integer variables are usually many bytes in size
- In our example program, they are 4-bytes in size.

(2) Strongly-typed programming languages are strict in distinguishing between characters and integers and will not allow the programmer to combine them in expression.

To illustrate the size difference between integers (32 bits) and characters, (8 bits) Consider the following program fragments:

```
ch = 241
ch = 242
ch = 243
ch = 244
ch = 245
ch = 246
ch = 247
ch = 248
ch = 249
ch = 250
ch = 251
ch = 252
ch = 253
ch = 254
ch = 255
ch = 0
ch = 1
ch = 2
ch = 3
ch = 4
ch = 5
ch = 6
ch = 7
```

```c
#include<stdio.h>

unsigned char ch;


int main(){
    printf("The values that ch can contain are: \n");
    for(ch = 0; ;ch++)
        printf("ch = %d\n", ch);
}
```

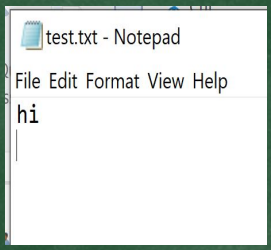
Bit Sequence overflows after $2^8 - 1$

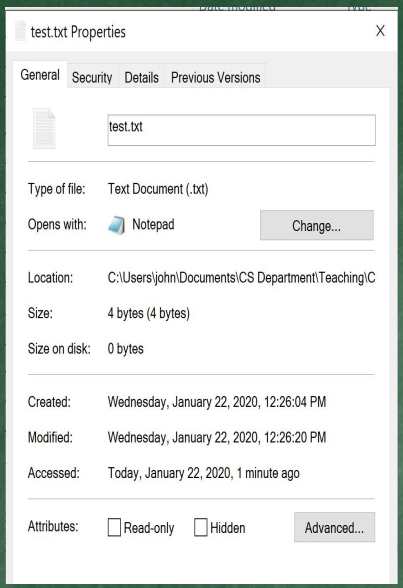
```c
#include<stdio.h>

#define LARGEINT 4294967000

unsigned int i;

int main(){
    printf("The values that i can contain are: \n");
    for(i = LARGEINT; ;i++)
        printf("i = %u\n", i);
}
```

```
i = 4294967284
i = 4294967285
i = 4294967286
i = 4294967287
i = 4294967288
i = 4294967289
i = 4294967290
i = 4294967291
i = 4294967292
i = 4294967293
i = 4294967294
i = 4294967295
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
i = 11
```

Bit Sequence overflows after $2^{32} - 1$.

How are Characters manipulated by Editors?

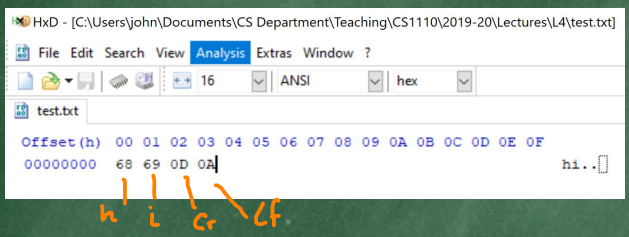




The 4 bytes in test.txt are:

'h' 'i' 'Cr' 'LF'

In Hex, the Contents of test.txt is: 68 69 0D 0A



HxD is a Hex Editor

Compare the size of the files generated by notepad and word for the same 'file contents'.

Representing characters by numbers is another level of abstraction.

Lots of Bytes

Multiples of bytes V·T·E

Decimal			Binary				
Value	Metric		Value	IEC		JEDEC	
1000	kB	kilobyte	1024	KiB	**kibibyte**	KB	kilobyte
1000^2	MB	megabyte	1024^2	MiB	mebibyte	MB	megabyte
1000^3	GB	gigabyte	1024^3	GiB	gibibyte	GB	gigabyte
1000^4	TB	terabyte	1024^4	TiB	tebibyte		–
1000^5	PB	petabyte	1024^5	PiB	pebibyte		–
1000^6	EB	exabyte	1024^6	EiB	exbibyte		–
1000^7	ZB	zettabyte	1024^7	ZiB	zebibyte		–
1000^8	YB	yottabyte	1024^8	YiB	yobibyte		–

Orders of magnitude of data

In this Course, we will Continue to use the older notation. Thus:

Kilo – K – 2^{10}
Mega – M – 2^{20}
Giga – G – 2^{30}
Tera – T – 2^{40}
Peta – P – 2^{50}
Exa – E – 2^{60}
Zetta – Z – 2^{70}
Yotta – Y – 2^{80}

1 YottaByte is close to the estimated number of atoms in a human Body.