An Introduction to Computers

Topics Covered

- Computer Logic
- Bits and Bytes
- Representing Numbers and Characters
  - Binary
  - Decimal
  - Hexadecimal
- Algorithms
- Instructing computers to perform complex tasks
- Code Reusability and Sharing

A brief introduction to Logic

- Logic is a science that sets forth rules for properly-ordered thinking
  - Helps with identifying faulty reasoning (fallacies)
  - Helps with verifying a claim's validity
    - The only true way to verify is to test!
- Critical Thinking is the application of those rules
  - Based on criteria that have been testing and verified
  - Logically combines those criteria to form an argument

Why talk about Logic

- Logic can be represented in any number of ways
  - Through words
  - Through math
  - Through code
- Computers fundamentally work through logic

How?

The Turing Machine

- One of the simplest ways of representing logic is through a simple switch.
  - *On* or *Off*. *Yes* or *No*. *True* or *False*. *1* or *0*.
- A Turing machine consists of a large number of these switches.
- For instance, let's think of a light that can have 4 states:
  - Off
  - Red Light
  - Yellow Light
  - Green Light
- How many switches would we need to use to produce these states?

Switches

- Two. Merely two. How can we do that?
- Let's call our switches Left and Right. Each can be Up or Down.
- So, here is an example of how we could match each state:
  - Off – Left Down, Right Down
  - Red Light – Left Down, Right Up
  - Blue Light – Left Up, Right Down
  - Yellow Light – Left Up, Right Up

A bit less wordy

- Okay, great, we were able to set up that light!
- That being said, out solution was a bit… Wordy.
- So, rather than saying Left or Right, let's just say Up or Down for each
- For Example:
  - Off – Down, Down
  - Red Light – Down, Up
  - Blue Light – Up, Down
  - Yellow Light – Up, Up

Can we go further?

- Alright, looking good!
- That being said, let's see if we make that *even shorter*
- Let's use 1 for Up and 0 for Down.
- Also, let's get rid of the comma.
- For Example:
  - Off – 00
  - Red – 01
  - Blue – 10
  - Yellow – 11

Even more complex

- Congratulations! What you just saw is Binary.
- That being said, what if we wanted to combine colors?
- For instance, combine Red and Yellow to get orange.

- In total we have these states:
  - Off
  - Red
  - Purple
  - Blue
  - Green
  - Yellow
  - Orange
  - Black
- How many switches would that take?

More Switches

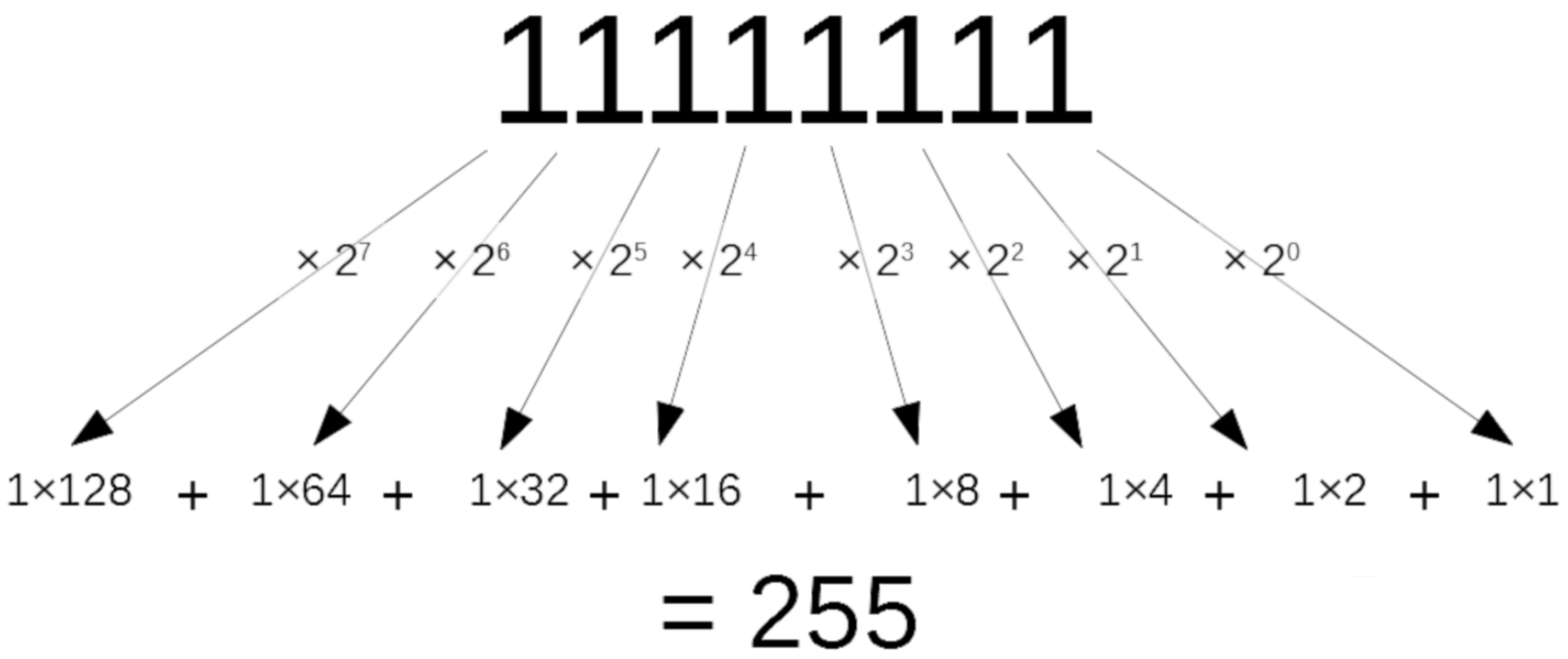
- One answer would be 3.
- In other words each switch represents a color.
- If a switch is off, that color is not being shown.

- For Example:
  - Off – 000
  - Red – 001
  - Blue – 010
  - Purple – 011
  - Yellow – 100
  - Orange - 101
  - Green - 110
  - Black - 111

Bits and Bytes

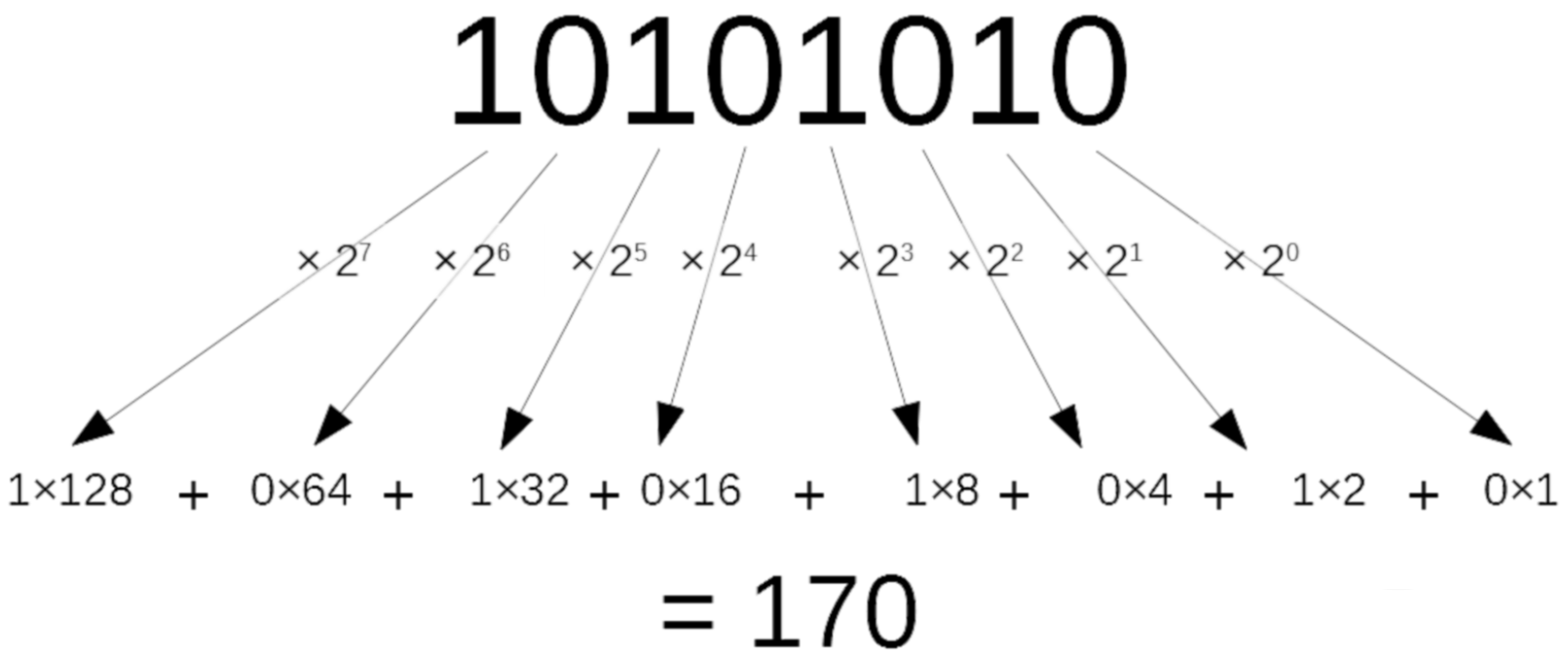
- In Computer Science, a Switch (0 or 1) is called a Bit
- So, our previous color scheme was represented by 3 Bits.
- A Byte, on the other hand, is represented by 8 bits.
  - For a total of 2^8 or 256 possible combinations
  - Byte is the smallest addressable unit of memory in most computer architectures.
  - Byte is the smallest variable size in high-level languages like C#, Java and Python.

Representing Numbers with Binary



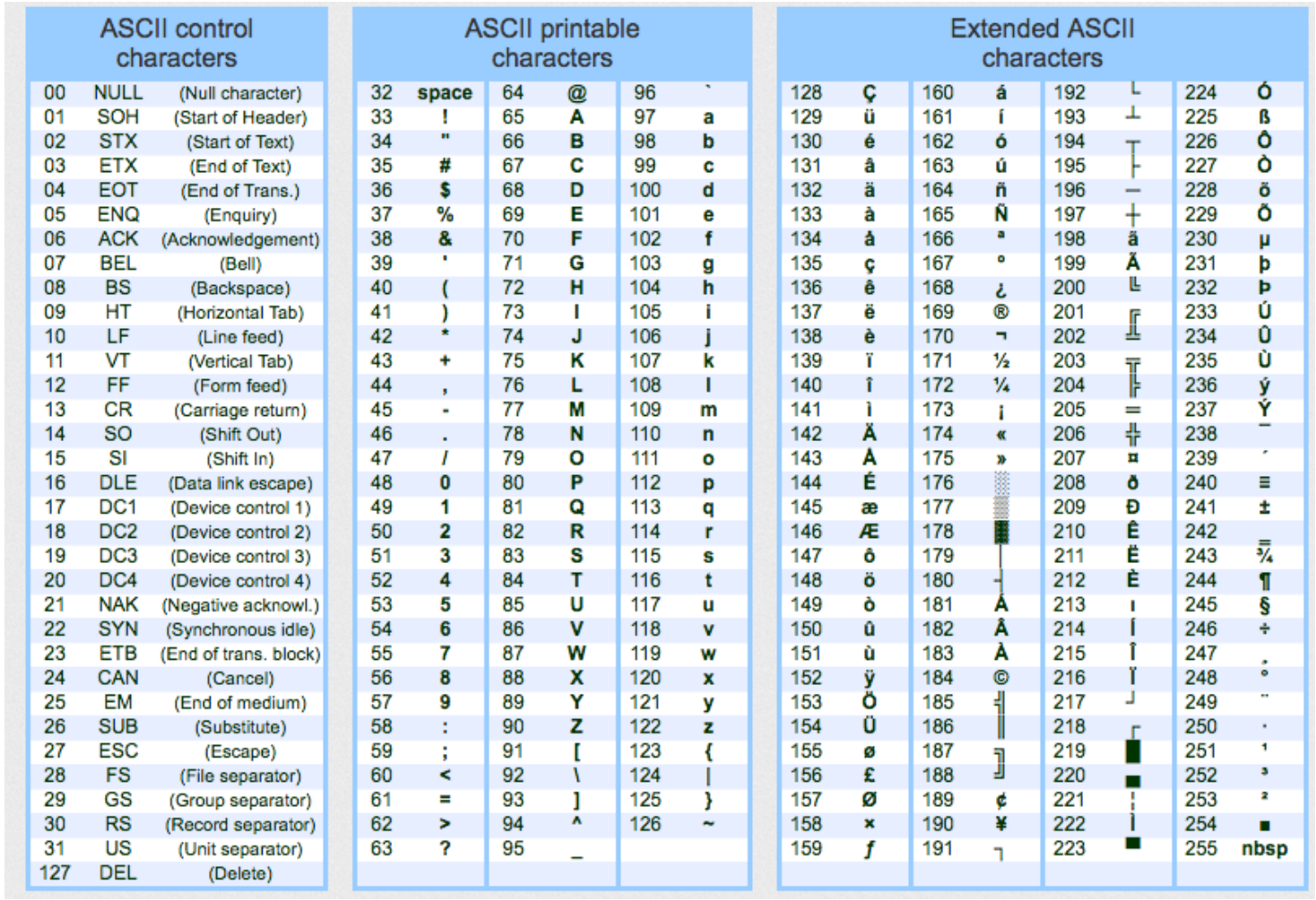
The maximum value we can represent with byte is 255. The minimum is 0. That is a total of 256 values.

Representing Numbers with Binary II



Every decimal value from 0 to 255 can be represent with a byte.

The ASCII Character Set



ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	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 acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

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	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	▒	208	ð	240	≡
145	æ	177	▓	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

ASCII stands for American Standard Code for Information Interchange.

ASCII was originally developed for use with teletype systems, then adapted for use with earlier computer systems.

Modern character encodings incorporate ASCII, but use 16 bits (2 bytes) and include many more characters – up to 65,536 (2^{16}).

Representing Characters and Digits

We can see from the previous slide that our common letters and digits are represented by numbers. For example, the digit 9 maps to the number 57. The letter Z maps to 90, and z maps to 122.

We are all accustomed to working with numbers using decimal, or “base 10” representation. Using base 10, we can represent all numbers using 10 digits, 0-9.

With binary or “base 2”, we can represent all numbers using two digits, 0 and 1.

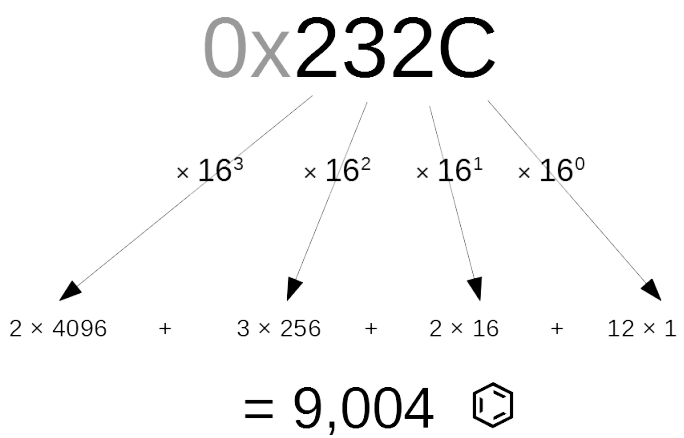
Computer scientists and programmers often use **hexadecimal**, or **base 16**, to represent numbers.

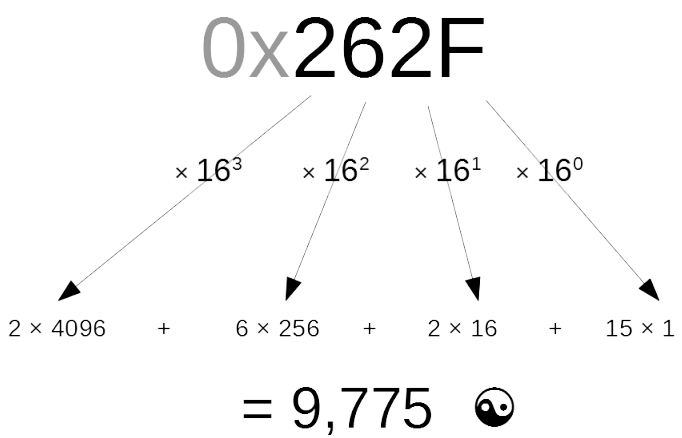
Character	Value (Decimal)	Value (Binary)
9	57	00111001
Z	90	01011010
z	122	01111010

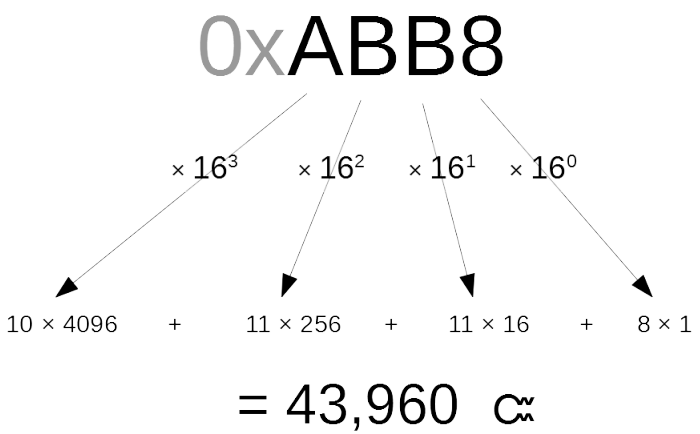
Hexadecimal Numbers

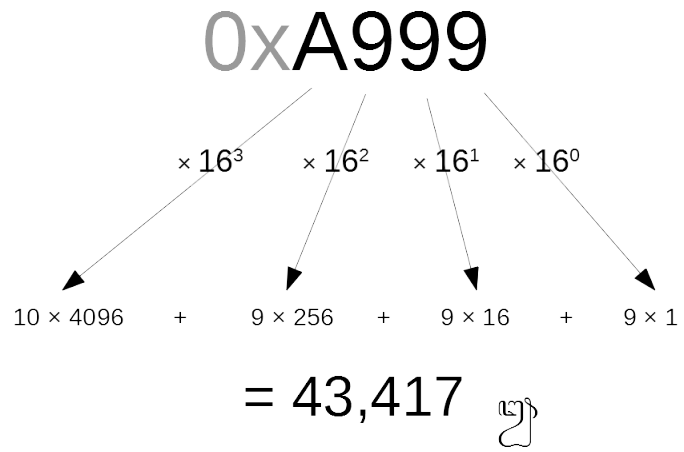
- Binary numbers allow us to see the byte structure, but they are difficult to read and write.
- When understanding the byte structure is important, computer scientists and programmers use hexadecimal.
- Hexadecimal uses digits 0-9 plus letters A-F to represent 10 – 15.
- Hexadecimal numbers are written with the prefix ‘0x’.
- The maximum value of a byte (255) is 0xFF.

Hexadecimal Numbers and Unicode









- The Unicode character set uses values from 0x0000 to 0xFFFF – that is 65,536 unique values.
- Using Unicode characters, computers can display information in most of the world's languages and alphabets.
- Unicode also includes many symbols and emojis.

Takeaways for Binary and Hex

- At their core, computers operate through many 0/1 switches.
- In order to represent their values more easily, we use hexadecimal.
- Through logic, combinations of these switches can be bound to different capabilities, such as storing numbers, displaying text, etc.
- Higher level programming languages are *compiled* – analyzed and converted into binary - allowing programmers to provide instructions to machines through program logic.

Algorithms

- If hexadecimal is how computers “think” and programming languages are how we communicate with them, *algorithms* are what we tell them to do.
- An algorithm is a repeatable set of steps designed to achieve a desired outcome.
- This process is meant to take certain inputs and consistently produce a set of outputs.
- For example:
  - Find the minimum value in an array of numbers
  - Read all of the text from a text file
  - Sort a list of names alphabetically

Find the minimum value in a list of numbers

- **Input**: A list of numbers
  - Step 1 – If the list is empty, generate an error: Empty list has no minimum value.
  - Step 2 – Define a variable named **min** and set it to the first value in the list.
  - Step 3 – Step through each value in the list and test whether it is less than **min**.
  - Step 3a: If the value is less than **min**, set min to that value.
  - Step 4 – Return **min**

Output: The minimum value in the list

Read all of the text from a text file

- **Input**: The path (directory and name) to a file
  - Step 1 – Test whether the file exists. If not, generate an error: File Not Found
  - Step 2 – Create an object which points to the location of that file on the disk drive.
  - Step 3 – Allocate memory for a buffer equal to the size of the file.
  - Step 4 – Read all bytes from the file into the buffer.
  - Step 5 – Close the file.
  - Step 6 – Use the file's encoding to convert the bytes in the buffer to text characters. ASCII encoding means that each byte represents one character; Unicode encoding means that every two bytes represents one character.

Output: A string variable representing the contents of the text file.

Algorithms are Unambiguous Instructions

- Computers operate through a series of binary instructions.
- Computers can do nothing unless provided a clear set of instructions.
- An Algorithm is a clear set of instructions which are designed to accomplish a specific task.

But WOW – some of the steps are complicated. Do we really need to write all of those instructions to read text from a file, find the minimum value in an array.

And sorting a list of strings alphabetically is even more complicated!

The answer is NO. You rarely need to write instructions at this level of detail.

Object Oriented Programming

- Modern programming languages allow us to create libraries of reusable code.
- Very smart developers often implement useful algorithms and then share their code with other developers.
- The developer community is generous like that!
- Using the appropriate C# libraries, finding the minimum value in an array is as simple as:
  - `double minValue = values.Min()`
- Reading all of the text from is file is as simple as:
  - `string text = File.ReadAllText(@"c:\books\mobydick.txt");`
- Sorting a list of strings is accomplished by:
  - `stringList.Sort();`

Topics Covered

- Computer Logic
- Bits and Bytes
- Representing Numbers and Characters
  - Binary
  - Decimal
  - Hexadecimal
- Algorithms
- Instructing computers to perform complex tasks
- Code Reusability and Sharing

Questions?