

CSC104 fall 2013  
Computational thinking  
week 6

Danny Heap  
heap@cs.toronto.edu  
BA4270 (behind elevators)

<http://www.cdf.toronto.edu/~heap/104/F12/>  
416-978-5899

Text: **Picturing Programs**

# Outline

More numbers

characters, images, sound

Really crude encryption

Notes

# Multiplication

multiply, shift, add

Once we can add non-negative integers, we can multiply them with a small number of additional operations. Consider the binary multiplication table:

×	0	1
0	0	0
1	0	1

I use this to multiply the binary representations of 15 and 19. Other arithmetic functions are implemented as particular circuits.

# Negative numbers, fractions

reassign some bits

Sometimes the left-most bit is used to represent + (as 0) or - (as 1). Another (efficient) scheme is called **two's complement**

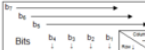
In base 10, by shifting a number right (past the decimal point) we multiply it by  $1/10$ . In binary, we shift right past the binary point, and reduce by  $1/2$ .

A common scheme, called **IEEE floating point**, uses 64 bits (binary digits): one for the sign, 11 for the magnitude (from  $2^{-1022}$  to  $2^{1023}$ ) and the remainder for an non-negative integer.

# Enough numbers!

what about text?

7 bits is enough to represent 128 values: upper- and lower-case latin characters, 10 numerals, some punctuation, and special control characters.



The diagram shows a 7-bit byte structure with bits labeled  $b_6$  through  $b_0$ . An arrow indicates the bit order from  $b_6$  to  $b_0$ . A small triangle labeled 'value' is positioned above the bit labels.

Bits	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$										
	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1
0	0	0	0	0	0	0	0	NUL	DLE	SP	0	@	P	'		p	
0	0	0	0	1	1	1	1	SOH	DC1	!	1	A	Q	a	q		
0	0	1	0	2	STX	DC2	*	2	B	R	b	r					
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s					
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t					
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u					
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v					
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w					
1	0	0	0	8	BS	CAN	(	8	H	X	h	x					
1	0	0	1	9	HT	EM	)	9	I	Y	i	y					
1	0	1	0	10	LF	SUB	*		J	Z	j	z					
1	0	1	1	11	VT	ESC	+		K	[	k	[					
1	1	0	0	12	FF	FC	.	<	L	\	l	l					
1	1	0	1	13	CR	GS	-	=	M	]	m	]					
1	1	1	0	14	SO	RS	.	>	N	^	n	^					
1	1	1	1	15	SI	US	/	?	O	_	o	_				DEL	

More than 110,000 characters can be specified with **unicode** (using more than 7 bits each).

# What about images

how do those pixels glow so?

Computers represent images as rectangular arrays of glowing pixels. There are various schemes to determine what colour each pixels glows, one is `rgba`, which is what we use in picturing-programs

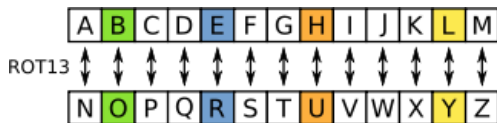
Each colour is a value between 0 and 255 (inclusive). This allows  $2^{24}$ , over 16 million colours. That's without even counting the "alpha" band, which represents opacity from clear (0) to opaque (255).

# Sound

All the complexity of a room full of instruments can be simulated using a stream of numbers. After all, you can model sound as the displacement of your eardrum one way or the other. That's what the **WAV sound format**, a variety of **LPCM** (Linear Pulse Code Modulation) does.

# a blast from the past

really bad text encryption



Encrypt "STRING"



## rot13 as an algorithm

- ▶ What is given, what's required?
- ▶ Redo the last step for a single character
- ▶ What is a really simple rule (or set of rules) for  $(\text{rot13 } c)$ , where  $c$  is some character?
- ▶ It might help to know that characters  $\#A$  through  $\#Z$  have ascii encodings 65 through 90.



## reversing strings

Give step-by-step instructions to reverse “string”

- ▶ given/required?
- ▶ check-expect some small examples?
- ▶ try to write down a recipe



# Notes

