CS102 Unit 0a – Course Intro

Mark Redekopp

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Introduction

- This is how we often see and interact with software
  - In truth we interact with it far more than we think
  - We are interacting with software when we drive, fly, turn on the lights, watch TV, go to the bank, or buy something with our credit card
- So what is it really?

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Introduction

- This is how the movies think computers see software
  - The far right picture is reasonably accurate

- While all programs eventually end up as 1s and 0s, we generally program using some form of "high-level" or scripting language

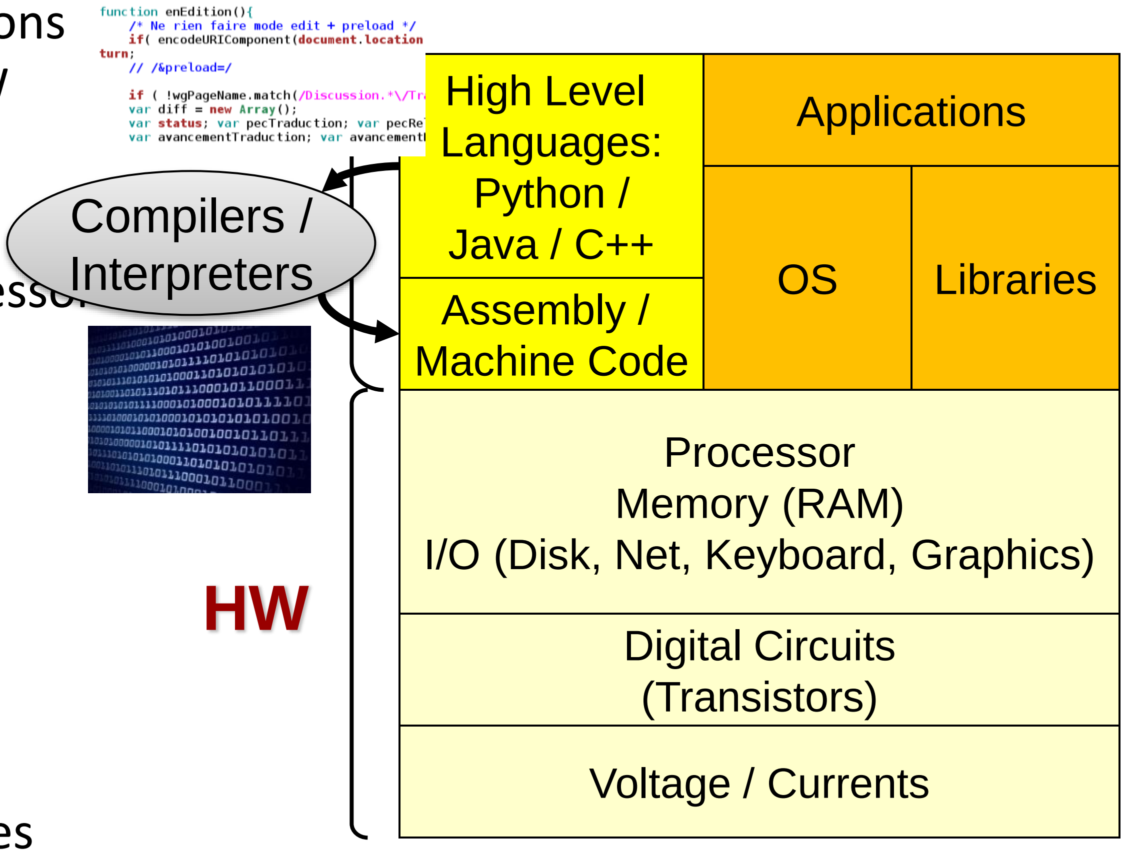
```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();          // display the fra
}
```

This Photo by Unknown Author is licensed under CC BY-SA

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Computer Abstractions

- Computer systems can be viewed as a layered stack of abstractions from basic HW to complex SW
- Assembly and machine code are the fundamental instructions a computer processor can execute
  - Too low level
- Enter high level languages
  - More powerful and succinct descriptive abilities
- Because of how the hardware works, our software must be written using certain structures
  - This class is intended to teach you those programming structures.



© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

This Class

- The goal of this class is two-fold
  - Teach you the basics of programming
  - Develop mathematical and algorithmic thinking skills needed to excel in future courses

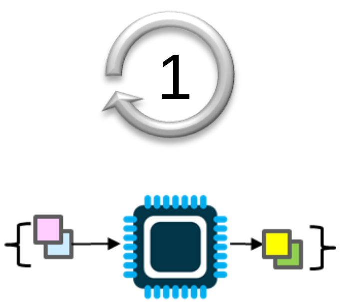
http://climbingla.blogspot.com/2010/05/walk-6-hermon-and-highland-park.html

© 2023 by Mark Redekopp. This content is protected and may...
http://epg.modot.org/index.php?title=234.2_Diamond_Interchanges

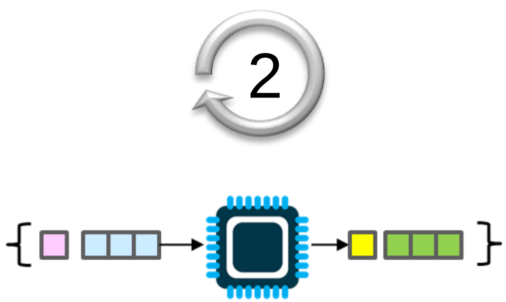
Course Structure

- The course is broken into 4 units each consisting of:

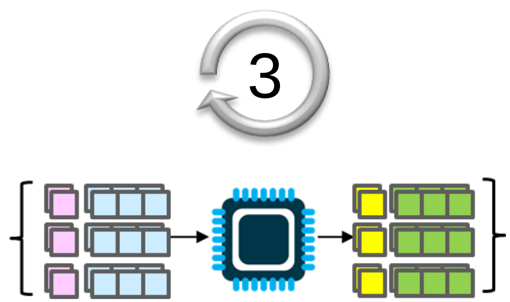
Lectures	Lab	Portfolio	Homework
	(Tools + Practice + Teamwork)	(Self-selected programming problems)	(Individual practice and programming assignments)



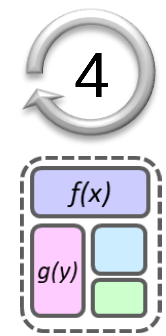
Scalar Processing (Expressions & Conditionals)



Linear (1D) Processing (Loops)



Multidimensional Processing (Nested Loops)



Divide & Conquer (Functions and Abstraction)

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

This Photo by Unknown Author is licensed under CC BY-NC

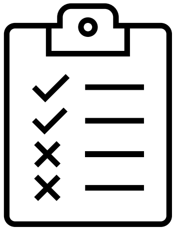
Exams and Grading

- The course will utilize 3 exams during our Quiz section

    Midterm 1 – Oct. 6

    Midterm 2 – Nov. 3

    Final – Dec. 9



- Grading will be as follows:

Labs	6%
Portfolio	6%
Homework	32%
Lowest Midterm	12%
Highest Midterm	22%
Final Exam	22%
Total	100%

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Syllabus

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Expectations

- Attend lectures & be engaged
  - Ask questions
  - We're a team…I need you!
  - I'll give you my best. Try to give me yours!
- Catch the wave!
  - Start assignments early, schedule weekly practice time, read and review other sources of input

This Photo by Unknown Author is licensed under CC BY-SA-NC

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

20-Second Timeout

- Who Am I?
  - Teaching faculty in EE and CS
  - Undergrad at USC in CECS
  - Grad at USC in EE
  - Work(ed) at Raytheon
  - Learning Spanish (and Chinese?)
  - Sports enthusiast!
    - Basketball
    - Baseball
    - Ultimate Frisbee?

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Programming Languages 1

- Declarative Languages
  - Describe the what but not the how
  - Examples: HTML, CSS

```html
<!DOCTYPE html>
<html>
    <head>My webpage</head>
    <body>
        <h1>Introduction</h1>
        <p>In a land far, far away...</p>
    </body>
</html>
```

```css
h4{
  font-size: 70px;
  font-family: Futura;
  position:absolute;
  top: 5%;
  left: 59%;
}
.number {
  font-family: Futura;
  font-size: 200px;
  position:absolute;
  top: 15%;
  left: 39%;
}
```

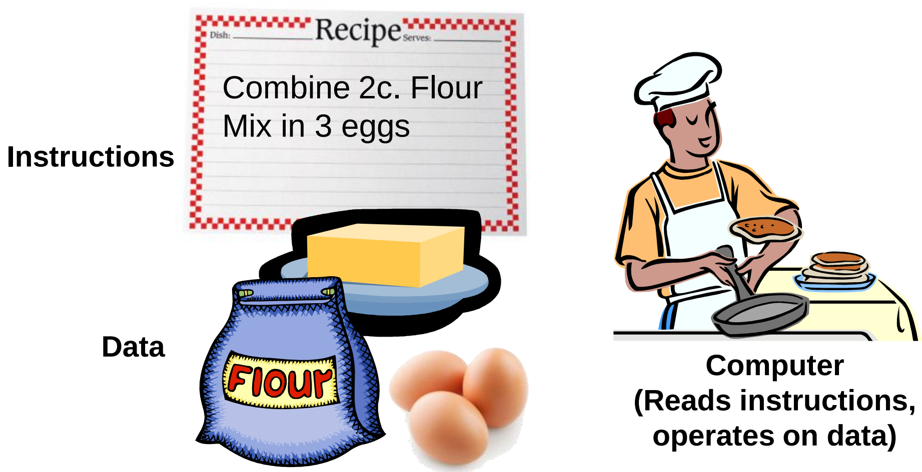
© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Programming Languages 2

- Imperative/Structured Languages
  - Describe the what (data) and how (instructions/algorithm)
  - Examples: C/C++, Java, Javascript, Python (which I'll use today)
  - The focus of most programming courses
  - Programs are like a recipe for how to operate on data

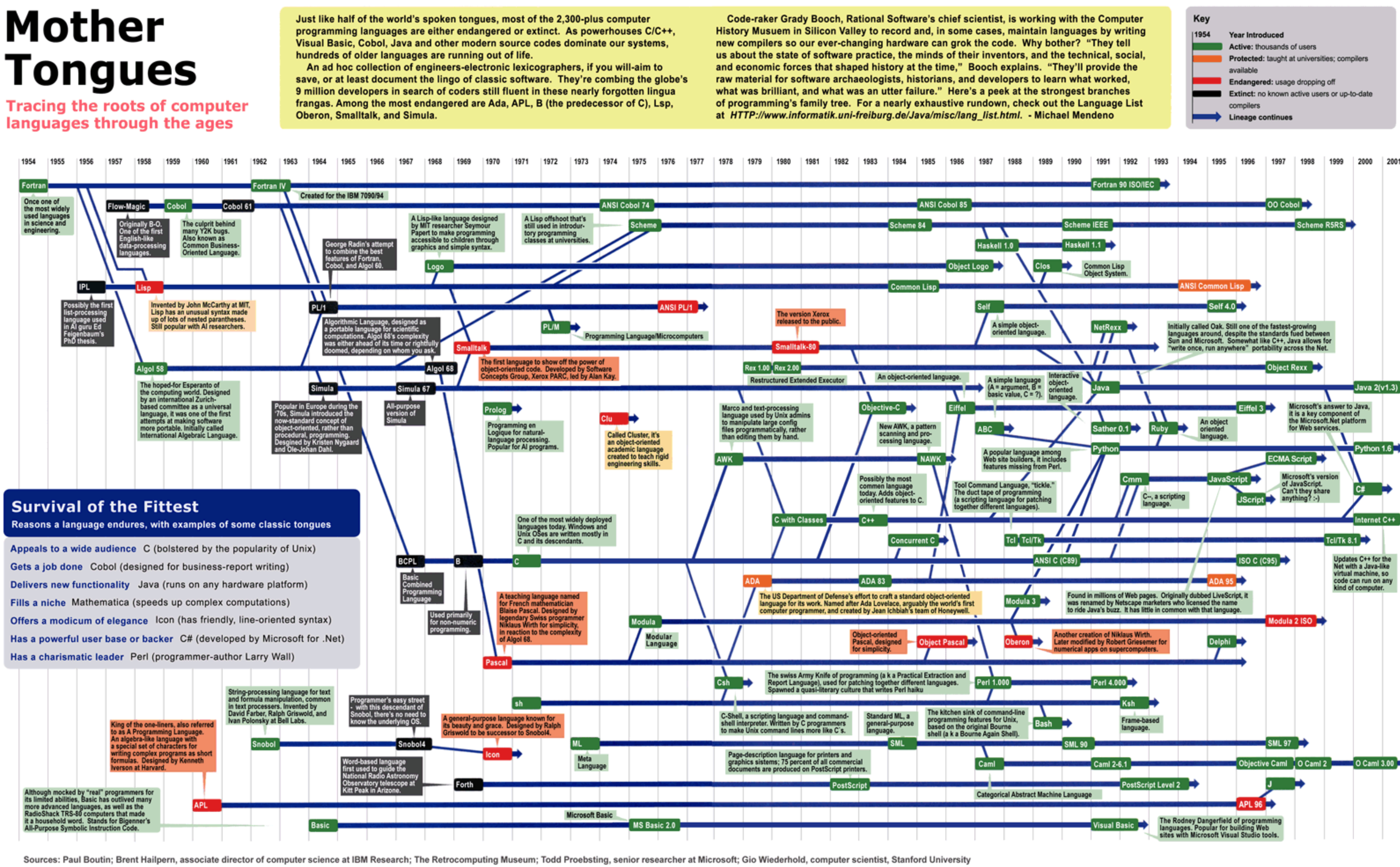
Quadratic Equation Solver

```python
import math

a = int(input("Enter a: "))
b = int(input("Enter b: "))
c = int(input("Enter c: "))

det = b*b - 4*a*c
if(det >= 0):
    r1 = (-b - math.sqrt(det)) / (2*a)
    r2 = (-b + math.sqrt(det)) / (2*a)
    print(f'Roots are {r1} and {r2}')
else:
    print("Imaginary roots")
```



© 2023 ... ted.

High Level Languages



http://www.digibarn.com/collections/posters/tongues/ComputerLanguagesChart-med.png

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Why C++

- C++ is used widely
- C++ is "close" to the hardware (HW)
  - Makes it fast
  - Makes it flexible (Near direct control of the HW)
  - Makes it dangerous (Near direct control of the HW)
  - In fact, many other languages are themselves written in C/C++
- Because if you learn C++ you can likely learn MOST languages very quickly
- Because that's what we use in CS 103

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

What Language Aspects Will We Learn?

- Programming skills in C/C++
  - Overlaps with the first 20% of CS 103
  - Data Representation
  - Basics of discrete mathematics
  - Expressions
  - Conditional Statements
  - Iterative Statements (Loops)
  - Functions
  - Arrays
- Problem solving using common programming **'idioms'**

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

More than just "Coding"...

Application

Language

Level	Description
Specification	• A precise problem statement to capture what the application requires (often requires the designer to make choices)
Problem Solving	• Understanding specification • Planning, especially partitioning into sub-problems • Identifying and using appropriate idioms • Solving difficult sub-problems • Writing "glue code" to tie everything together
Idioms	• Simple programming patterns/templates for solving specific tasks that can be used to connect your problem solving approach to actual code
Semantics	• Meaning of a program or any of its parts
Syntax	• Rules/grammar of the language

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Problem Solving Idioms

- An idiom is a colloquial or common mode of expression
  - Example: "raining cats and dogs"
- Programming has common modes of expression that are used quite often to solve problems algorithmically
- We have developed a repository of these common programming idioms. We STRONGLY suggest you…
  - Reference them when attempting to solve programming problems
  - Familiarize yourself with them and their structure as we cite them until you feel comfortable identifying them

Rule / Exception Idiom

- **Name** : Rule/Exception
- **Description** : Perform a default action and then us an `if` to corre
- **Structure**: Code for some default action (i.e. the rule) is followed b exceptional case

```
// Default action

if( /* Exceptional Case */ )
{
    // Code for exceptional case
}
```

  - **Example(s)**:
  - Base pay plus bonus for certain exceptional employees

```
bool earnedBonus = /* set somehow */;
int bonus = /* set somehow */;

int basePay = 100;
if( earnedBonus == true )
{
    basePay += bonus;
}
```

  - **Notes**: This can be implemented with an `if/else` where an else implements the other.

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

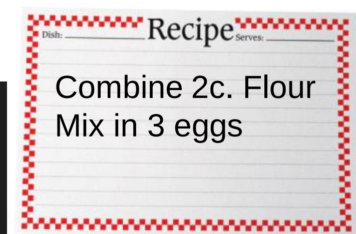
STARTING TO THINK LIKE A COMPUTER

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

It’s A Numbers Game

- Fact 1: Everything in a computer is a **number**
  - Sure. Things like 102 and 3.9 are numbers
  - But what about text and images and sound?
  - Everything!
- Fact 2: Computers can only work with or "see" 1 or 2 numbers at a time (i.e. they can only do 1 thing at a time)
- Humans process information differently
  - Therein lies some of the difficultly of learning programming

```
import math

a = int(input("Enter a: "))
b = int(input("Enter b: "))
c = int(input("Enter c: "))

det = b*b - 4*a*c
if(det >= 0):
    r1 = (-b - math.sqrt(det)) / (2*a)
    r2 = (-b + math.sqrt(det)) / (2*a)
    print(f'Roots are {r1} and {r2}')
else:
    print("Imaginary roots")
```



© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Example (1)

- What do you see?
  - The letter 'a'!

a

- What does the computer see?
  - A number; each text character is coded to a number
    - Example: Character map / Insert symbol

97

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

USC Viterbi School of Engineering

Text Representation

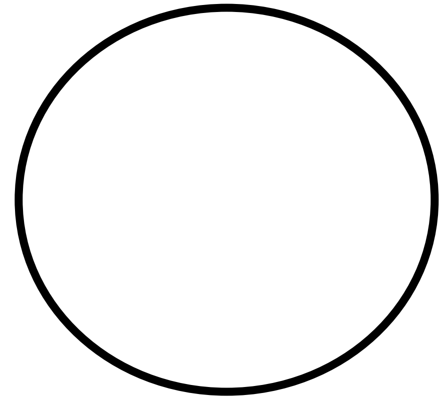
- Most common character code is ASCII (UTF-8)
- Every character, even non-printing, characters have a corresponding numbers
  - Decimal (base 10) / Hexadecimal (base 16)

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

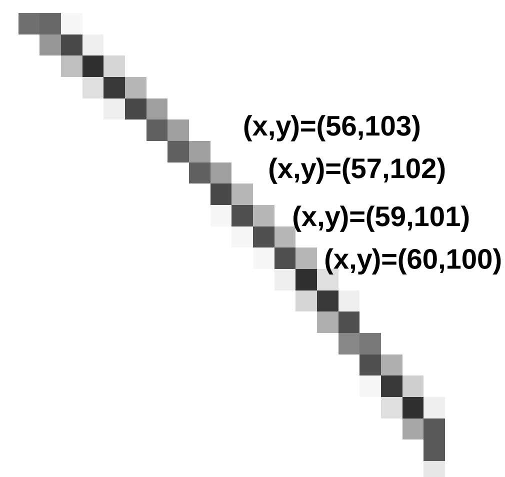
https://www.commfront.com/pages/ascii-chart

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Example (2)

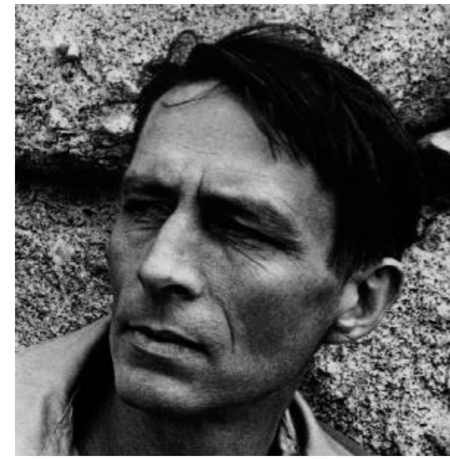


- What do you see?
  - A circle!
- What does the computer see?
  - Coordinate pairs of each "pixel"
  - ...or...
  - r = 120; origin = (10, 14)
  - Computer has to enumerate and visit each location and color it black

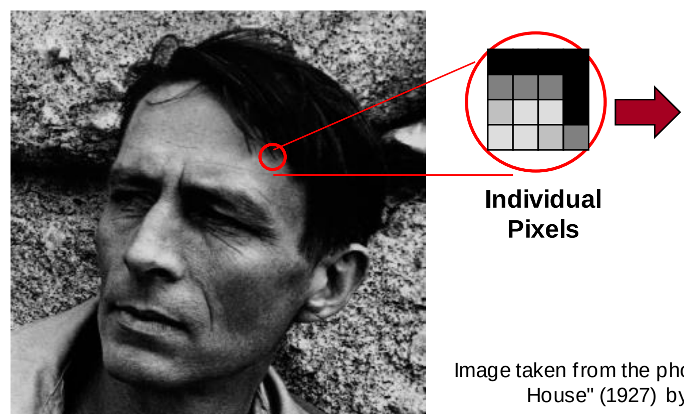


© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Example (3)



- What do you see?
  - A man's face!
- What does the computer see?
  - Many numbers (aka pixels)
  - Value corresponds to color



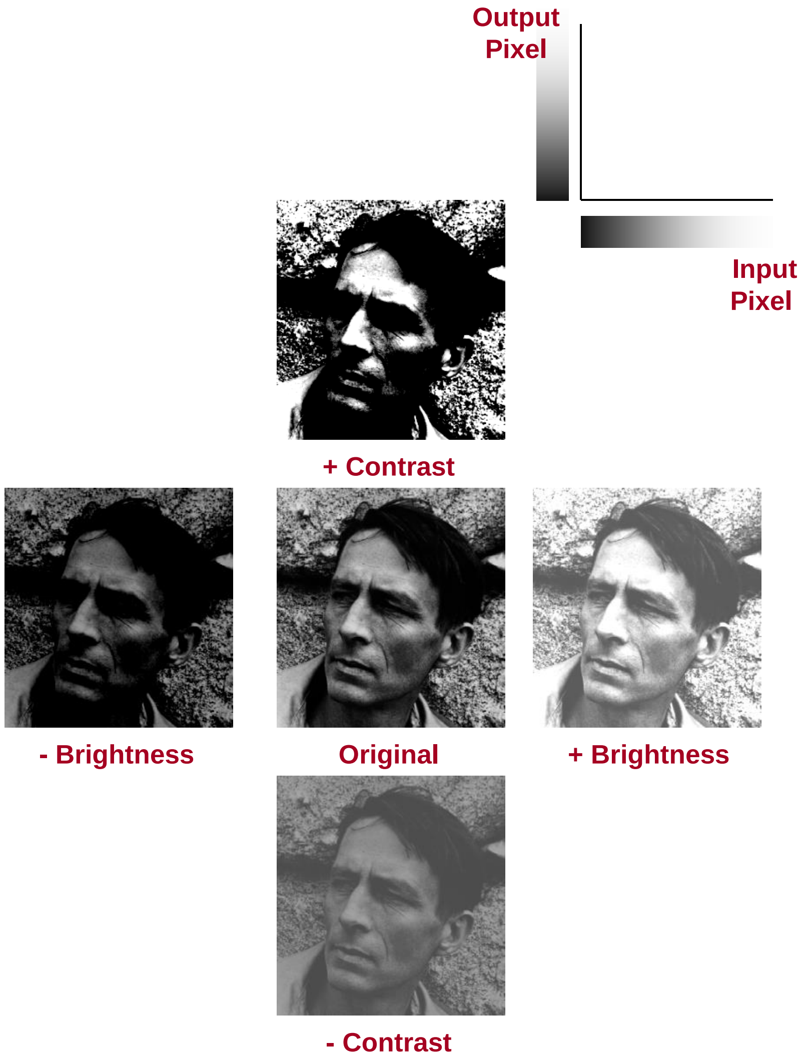
0	0	0	0
64	64	64	0
128	192	192	0
192	192	128	64

Image taken from the photo "Robin Jeffers at Ton House" (1927) by Edward Weston

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

The Connection with Mathematics

- Brightness
  - Each pixel value is increased/decreased by a constant amount
  - $P_{new} = P_{old} + B$
    - $B > 0$ = brighter
    - $B < 0$ = less bright
- Contrast
  - Each pixel value is multiplied by a constant amount
  - $P_{new} = C*P_{old} + k$
    - $C > 1$ = more contrast
    - $0 < C < 1$ = less contrast
- Same operations performed on all pixels



© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

"Enough" is NOT enough

- As we program we must be explicit
  - Example: drawing the circle on the screen
- Being general is not sufficient; we must be explicit!
  - Imagine a recipe for cinnamon rolls that simply read:
    - Mix and bake the following: butter, that white powdery baking substance, eggs, just **enough** sugar, and cinnamon. Enjoy!
  - How much of each, how much is "enough", how long, in what order?
- We will try to work on some of discrete math skills that help us explicitly define and analyze our programs

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Integers and floating-point types; Division and modulus operations

WORKING WITH NUMBERS IN C++

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Data Types

- How should the numbers (actually the bits: 1s and 0s) the computer is storing be interpreted: as a **letter**, an **integer** (aka an '**int**'), a number with decimals (aka **'floating point'** or **'double'**)
- C/C++ **types** help tell the computer hardware how to **interpret** the bits/numbers being stored in computer memory and what circuits to use to process them
- Let's learn the first two C++ data types:
  - `int` – integers only; no decimals (e.g. `4750, -18, 1908734`)
  - `double` – very large numbers all the way down to very small fractions (e.g. `6.02E23, 1.5, -0.000248`)

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Division

- Computers perform division differently based on the **types** used as inputs
- **Integer** Division:
  - When dividing two integer values, the result will also be an integer (any remainder/fraction will be dropped)
  - 10 / 4 = 2 52 / 10 = 5 6 / 7 = 0
- **Floating-point** (Double) & Mixed Division
  - 10.0 / 4.0 = 2.5 52.0 / 10 = 5.2 6 / 7.0 = 0.8571
  - Note: If one input is a double, the other will be promoted temporarily (aka ***implicitly "casted"***) to compute the result as a double

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Modulus

- Dividing two **integers** yields an integer quotient
- Using the modulus operator (%) will divide two **integers** but yield the remainder!
- Examples:

```
7 / 3 = 2        but 7 % 3 = 1
75 / 10 = 7      but 75 % 10 = 5
27 / 4 = __      but 27 % 4 = __
59 / 12 = __     but 59 % 12 = __
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

HOMEWORK AND SURVEY

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Modulus

- Dividing two integers yields an integer quotient
- Using the modulus operator (%) will divide two numbers but yield the remainder!
- Examples:

```
7 / 3 = 2        but 7 % 3 = 1
75 / 10 = 7      but 75 % 10 = 5
27 / 4 = 6       but 27 % 4 = 3
59 / 12 = 4      but 59 % 12 = 11
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.