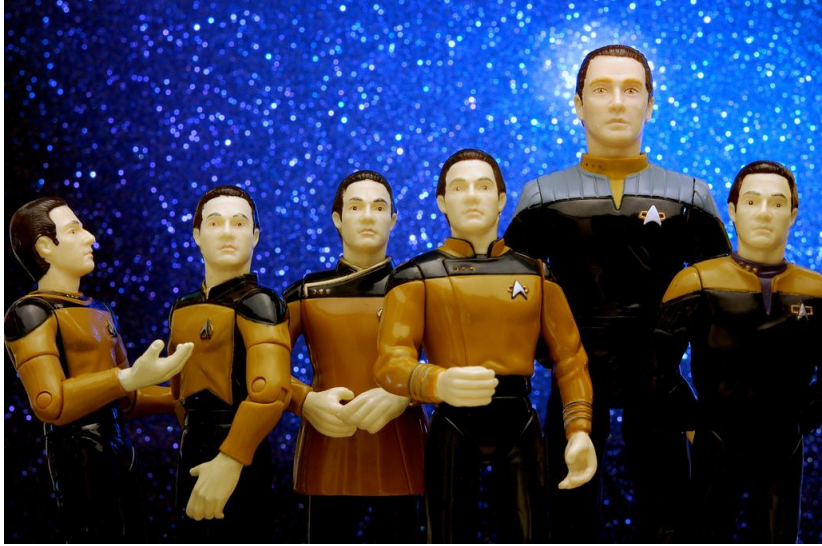
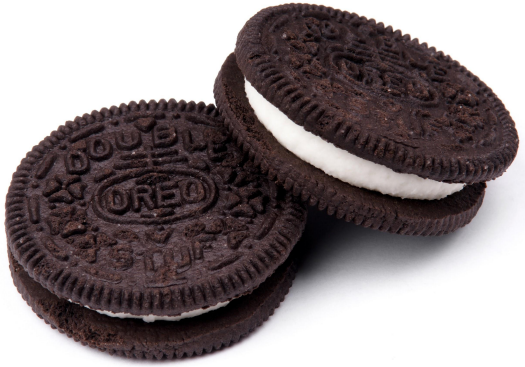
Data Representation

Data Types in C

- 4 Basic Types
  - Double
  - Float - already know
  - Int - already know
  - Char - already know



Double



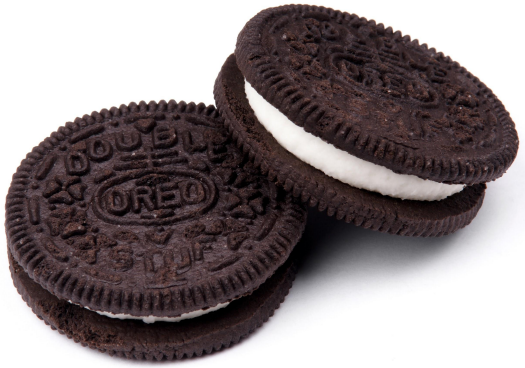
- Decimal value
- Uses 64 bits of memory when being stored
  - 52 bits → reserved for number
  - 11 bits → reserved for exponent, to store decimal
  - 1 bit → reserved for sign (positive / negative)

Double

- Largest possible value:
  - $2^{53} - 1$
  - *or* 9,007,199,254,740,991

- Smallest possible value:
  - Same thing as max value, just make it negative
  - $-2^{53} - 1$
  - *or* -9,007,199,254,740,991

Double vs Float



- Double has 64-bit precision
- Float has 32-bit precision
  - 23 bits reserved for number
  - 8 bits reserved for decimal
  - 1 bit reserved for sign

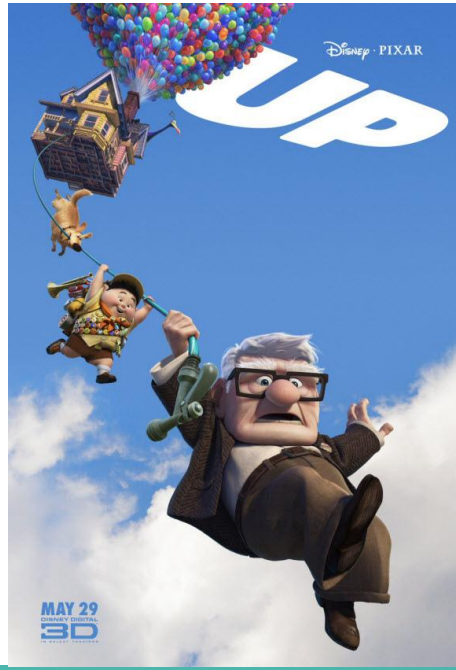
Precision

- Describes what values can be stored in certain variable types
- Double: (2^{53} - 1); with decimals
- Float: (2^{24} -1); with decimals
- Integer: (2^{32} - 1); no decimals
- Char: (2^{7} - 1); no decimals
- Boolean: 1 - 0; no decimals

Upcasting

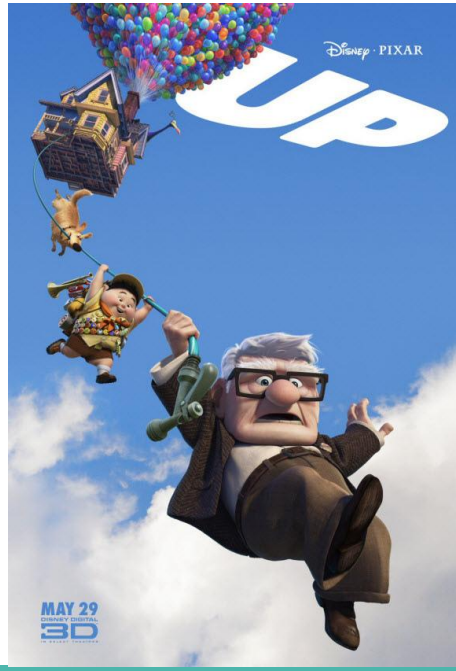
- Upcast → the ability to convert from a smaller data type into a larger data type

```
char foo = 42;

int bar = foo;

printf("Foo is: %c\n", foo);

printf("Bar is: %d\n", bar);
```



Upcasting

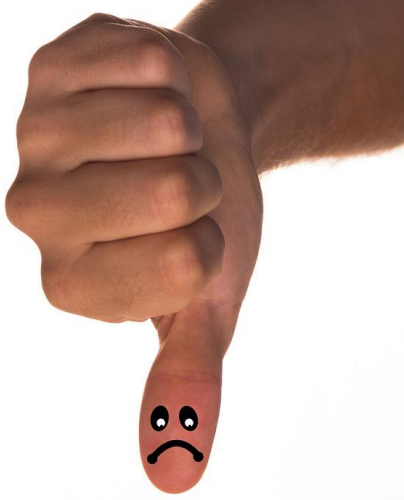
- ALWAYS SAFE to go from a smaller data type to a larger one
  - No loss of information
  - Try it out: Write a program that scans in an integer, then upcast it to a float. Print out both numbers



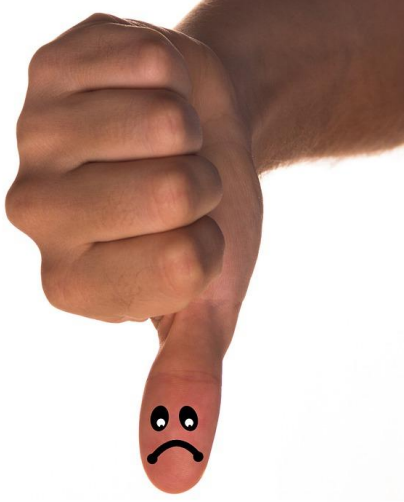
Downcasting

- Going from a larger data type to a smaller one
  - Ex: From integer to char

- DANGEROUS because information can be lost, without any warnings



Downcasting



- DANGEROUS because information can be lost, without any warnings

- Try it out: Change your upcast code to scan in a larger data type, then downcast to a smaller one. What problems do you run into?

Cast operator

- Can use the *cast* operator to implicitly change between data types

int meme = 42;

char pepe = **(char)** meme;

Cast operator

- Can use the *cast* operator to implicitly change between data types

- Upcast / downcast rules still apply

Practical uses of upcasting/downcasting

- Integer / Floating point division

- Review
  - Integer division causes decimal points to be cut off in final answer
  - Floating point division allows decimal points to stay

Practical uses of upcasting/downcasting

- Integer / Floating point division
  - Can *cast* an integer into a float in order to ensure floating point division occurs

```
int i = 42; int j = 5;

float answer = (float) i / j;
```

Coding Challenge

Scan in 5 (five) integers describing test scores in Programming. Find the average of these five scores and print it out with 2 (two) decimal place precision.