CS 106B

Lecture 28: Different Languages

Wednesday, August 16, 2017

Programming Abstractions
Summer 2017
Stanford University
Computer Science Department

Lecturer: Chris Gregg

reading:
Programming Abstractions in C++, Chapter 19

Today's Topics

- Logistics
  - Final Exam prep online: http://web.stanford.edu/class/cs106b/handouts/final.html
  - Mini-review session on Inheritance: STLC 111, 7:30pm, Wednesday
  - Final Review Session: STLC 111, 6:30-7:30pm, Thursday August 17th
  - Final exam is on Saturday, August 19th at 8:30am — I need a good list of students who will need power strip access; please fill out the survey on Piazza: https://piazza.com/class/j44l3eyaz006uz

- Different Languages You Might Like
  - PDF for code: http://web.stanford.edu/class/cs106b/lectures/28-DifferentLanguages/code/handout.pdf
  - Fun Languages
  - Python
  - Javascript and D3
  - Haskell
  - COBOL
  - Obfuscated C
  - Quines

Different Languages

There are literally *hundreds* of programming languages.

You probably have never heard of most of them, and you will almost certainly never program in most of them.

https://en.wikipedia.org/wiki/List_of_programming_languages

Some fun examples

There are also a number of "Esoteric Programming Languages," which can be fun:

https://en.wikipedia.org/wiki/Esoteric_programming_language

Some fun examples: LOLCODE

```
HAI 1.0
CAN HAS STDIO?
I HAS A VAR
IM IN YR LOOP
    UP VAR!!1
    VISIBLE VAR
    IZ VAR BIGGER THAN 10? KTHX
IM OUTTA YR LOOP
KTHXBYE
```

https://en.wikipedia.org/wiki/LOLCODE

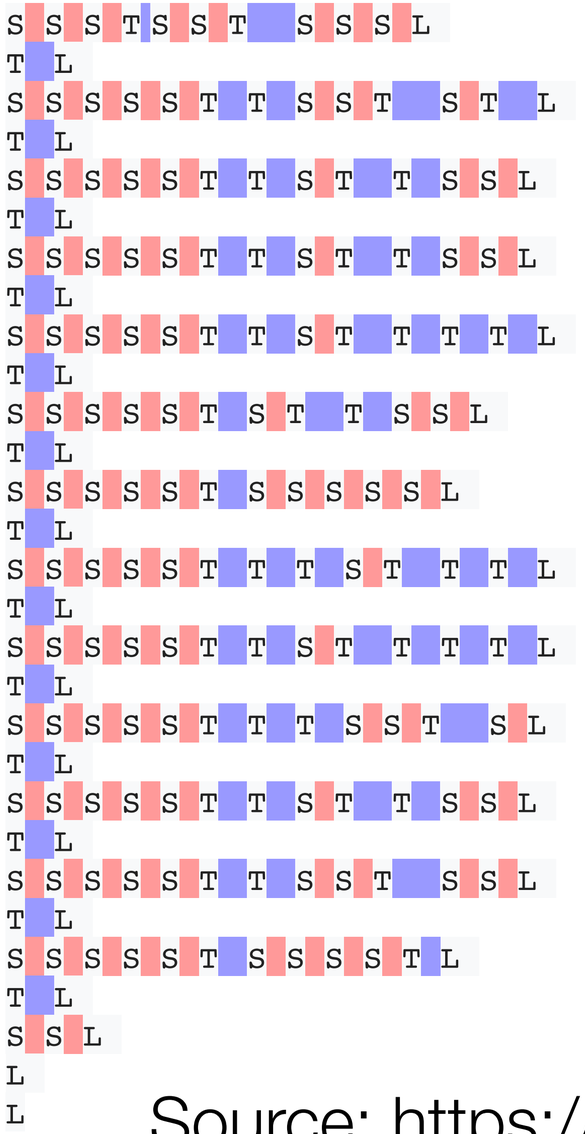
Some fun examples: Piet

A "Hello World" program in Piet

Piet is a language designed by David Morgan-Mar, whose programs are bitmaps that look like abstract art. The compilation is guided by a "pointer" that moves around the image, from one continuous coloured region to the next. Procedures are carried through when the pointer exits a region.

Source: https://en.wikipedia.org/wiki/Esoteric_programming_language#Piet

Some fun examples: Whitespace



A "Hello World" program in Whitespace

This is a commented Whitespace program that simply prints "Hello, world!", where each Space, Tab, or Linefeed character is preceded by the identifying comment "S", "T", or "L", respectively:

Source: https://en.wikipedia.org/wiki/Whitespace_(programming_language)

Python 

Python is an extremely readable programming language (it looks like psuedocode), and is really simple to learn (though it takes a long time to learn it really well!)

In Python, whitespace matters, but that means that curly braces aren't necessary:

```
for i in range(10):
    print("Hello " + str(i))
```

One cool feature of Python is that it has a "REPL" that you can simply open up and type on. Let's try it!

Python 

Python has some neat built-in functionality, such as arbitrary length integers:

```
>>> 2 ** 2 ** 2 ** 2 ** 2
```

The "**" means exponent. How many digits is this going to be?

Python 

Python has some neat built-in functionality, such as arbitrary length integers:

The "**" means exponent. How many digits is this going to be?

```
>>> len(str(2 ** 2 ** 2 ** 2 ** 2))
19729
```

Python 

Python has built-in lists, dictionaries (hash tables), sets, queues, psuedo-random number generators, etc., and it is also object oriented and has classes.

You can return multiple values in Python:

```
import math
def quadEqSolver(a,b,c):
    inner = math.sqrt(b * b - 4 * a * c)
    return (-b + inner) / (2 * a), (-b - inner) / (2 * a)
```

FizzBuzz

A famous easy coding interview question is "FizzBuzz," defined as follows:

Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

We can write this relatively easily in most languages, so we'll do it in the different languages we look at today.

C++ FizzBuzz

```
#include <iostream>
using namespace std;
int main ()
{
  for(int i = 1; i <= 100; i++)
  {
    if(i % 3 == 0 && i % 5 == 0) {
      cout << "FizzBuzz" << endl;
    } else if(i % 3 == 0) {
          cout << "Fizz" << endl;
    } else if(i % 5 == 0) {
          cout << "Buzz" << endl;
    } else {
        cout << i << endl;
    }
  }
  return 0;
}
```

Output:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
```

etc.

Python FizzBuzz

```
for i in range(1,100):
    if i % 3 == 0 and i % 5 == 0:
        print("FizzBuzz")
    elif i % 3 == 0:
        print("Fizz")
    elif i % 5 == 0:
        print("Buzz")
    else:
        print(i)
```

Output:

```
$ python fizzBuzz.py
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
```

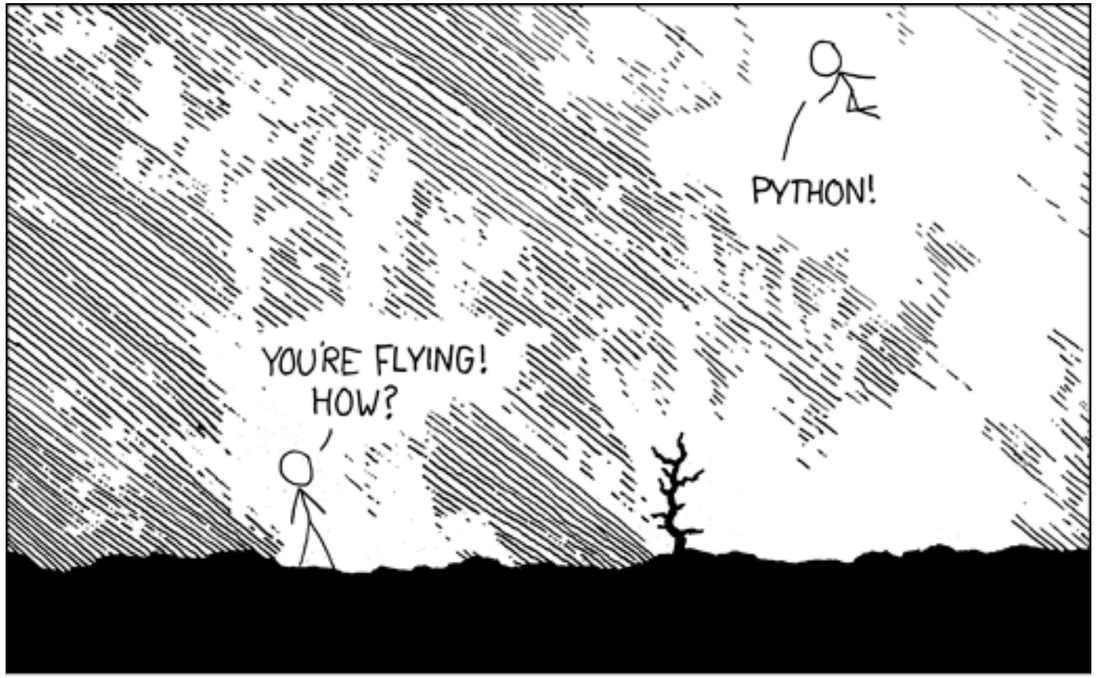
etc.

Python

If you are going to learn one more language, learn Python — you won't be disappointed!

```
>>> import this
```

```
>>> import antigravity
```



JavaScript and D3

JavaScript has become the "language of the Web"

It is not the best language, but it is a pretty good one (but, give the Author, Brendan Each, a break — he created it in 10 days!)

JavaScript and Java are completely different languages, although they both share "C-like syntax". So, for loops look very familiar:

```
for (var i=0; i < 10; i++) {
    console.log("Hello " + i);
}
```

You already have JavaScript installed and ready to go — it's built into the browser. Let's play!

JavaScript FizzBuzz

```
function fizzBuzz() {
    for(var i = 1; i <= 100; i++)
    {
        if(i % 3 == 0 && i % 5 == 0) {
            console.log("FizzBuzz");
        } else if(i % 3 == 0) {
            console.log("Fizz");
        } else if(i % 5 == 0) {
            console.log("Buzz");
        } else {
            console.log(i);
        }
    }
}
```

Looks a lot like C++!

JavaScript syntax is basically the same, but the model is much different!

JavaScript FizzBuzz, output to Web Page

FizzBuzz.html:

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript"
          src="fizzBuzz_web.js"></script>
    </head>
    <body onload=fizzBuzz()>
        <div id="fizzBuzzOutput">
            This is the default text.
        </div>
    </body>
</html>
```

fizzBuzz_web.js:

```
function fizzBuzz() {
    fb_div = document.getElementById('fizzBuzzOutput');

    // clear current contents
    fb_div.innerHTML = ""

    for(var i = 1; i <= 100; i++)
    {
        if(i % 3 == 0 && i % 5 == 0) {
            fb_div.innerHTML += "FizzBuzz<br>";
        } else if(i % 3 == 0) {
            fb_div.innerHTML += "Fizz<br>";
        } else if(i % 5 == 0) {
            fb_div.innerHTML += "Buzz<br>";
        } else {
            fb_div.innerHTML += i + "<br>";
        }
    }
    return 0;
}
```

JavaScript FizzBuzz, output to Web Page

FizzBuzz.html:

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript"
          src="fizzBuzz_web.js"></script>
    </head>
    <body onload=fizzBuzz()>
        <div id="fizzBuzzOutput">
            This is the default text.
        </div>
    </body>
</html>
```

fizzBuzz_web.js:

```
function fizzBuzz() {
    fb_div = document.getElementById('fizzBuzzOutput');

    // clear current contents
    fb_div.innerHTML = ""

    for(var i = 1; i <= 100; i++)
    {
        if(i % 3 == 0 && i % 5 == 0) {
            fb_div.innerHTML += "FizzBuzz<br>";
        } else if(i % 3 == 0) {
            fb_div.innerHTML += "Fizz<br>";
        } else if(i % 5 == 0) {
            fb_div.innerHTML += "Buzz<br>";
        } else {
            fb_div.innerHTML += i + "<br>";
        }
    }
    return 0;
}
```

One problem…we want our websites to be responsive, and we don't like loops!

JavaScript FizzBuzz, output to Web Page

fizzBuzz_web_noLoop.js:

Set a timer for each iteration

Fires every millisecond

```
function fizzBuzz() {
    fb_div = document.getElementById('fizzBuzzOutput');

    // clear current contents
    fb_div.innerHTML = ""

    var i = 1;
    var timer = setInterval(function() {
        if(i % 3 == 0 && i % 5 == 0) {
            fb_div.innerHTML += "FizzBuzz<br>";
        } else if(i % 3 == 0) {
            fb_div.innerHTML += "Fizz<br>";
        } else if(i % 5 == 0) {
            fb_div.innerHTML += "Buzz<br>";
        } else {
            fb_div.innerHTML += i + "<br>";
        }
        i++;
        if (i == 100) {
            // stop the timer
            clearInterval(timer);
        }
    },1);
    return 0;
}
```

JavaScript FizzBuzz, output to Web Page

fizzBuzz_web_noLoop.js:

This is called an "anonymous function", and it is what gets called every time the timer fires.

It takes some getting used to!

```
function fizzBuzz() {
    fb_div = document.getElementById('fizzBuzzOutput');

    // clear current contents
    fb_div.innerHTML = ""

    var i = 1;
    var timer = setInterval(function() {
        if(i % 3 == 0 && i % 5 == 0) {
            fb_div.innerHTML += "FizzBuzz<br>";
        } else if(i % 3 == 0) {
            fb_div.innerHTML += "Fizz<br>";
        } else if(i % 5 == 0) {
            fb_div.innerHTML += "Buzz<br>";
        } else {
            fb_div.innerHTML += i + "<br>";
        }
        i++;
        if (i == 100) {
            // stop the timer
            clearInterval(timer);
        }
    },1);
    return 0;
}
```

JavaScript FizzBuzz, output to Web Page

fizzBuzz_web_noLoop.js:

```
function fizzBuzz() {
    fb_div = document.getElementById('fizzBuzzOutput');

    // clear current contents
    fb_div.innerHTML = ""

    var i = 1;
    var timer = setInterval(function() {
        if(i % 3 == 0 && i % 5 == 0) {
            fb_div.innerHTML += "FizzBuzz<br>";
        } else if(i % 3 == 0) {
            fb_div.innerHTML += "Fizz<br>";
        } else if(i % 5 == 0) {
            fb_div.innerHTML += "Buzz<br>";
        } else {
            fb_div.innerHTML += i + "<br>";
        }
        i++;
        if (i == 100) {
            // stop the timer
            clearInterval(timer);
        }
    },1);
    return 0;
}
```

Increment `i` to keep track of the iteration count

JavaScript FizzBuzz, output to Web Page

fizzBuzz_web_noLoop.js:

```
function fizzBuzz() {
    fb_div = document.getElementById('fizzBuzzOutput');

    // clear current contents
    fb_div.innerHTML = ""

    var i = 1;
    var timer = setInterval(function() {
        if(i % 3 == 0 && i % 5 == 0) {
            fb_div.innerHTML += "FizzBuzz<br>";
        } else if(i % 3 == 0) {
            fb_div.innerHTML += "Fizz<br>";
        } else if(i % 5 == 0) {
            fb_div.innerHTML += "Buzz<br>";
        } else {
            fb_div.innerHTML += i + "<br>";
        }
        i++;
        if (i == 100) {
            // stop the timer
            clearInterval(timer);
        }
    },1);
    return 0;
}
```

Stop the timer after we reach 100

D3.js

If you are taking cs193c this quarter, you already know a bit about JavaScript, and HTML. There are a ton of JavaScript libraries online. One of my favorites is a JavaScript library that can make dynamic data visualizations really easily.

Let's look at some examples! 

http://christopheviau.com/d3list/gallery.html

https://github.com/d3/d3/wiki/Gallery

eclipse.js

Yesterday, I wrote a little D3 animation to show the path of the Solar Eclipse that is happening next week.

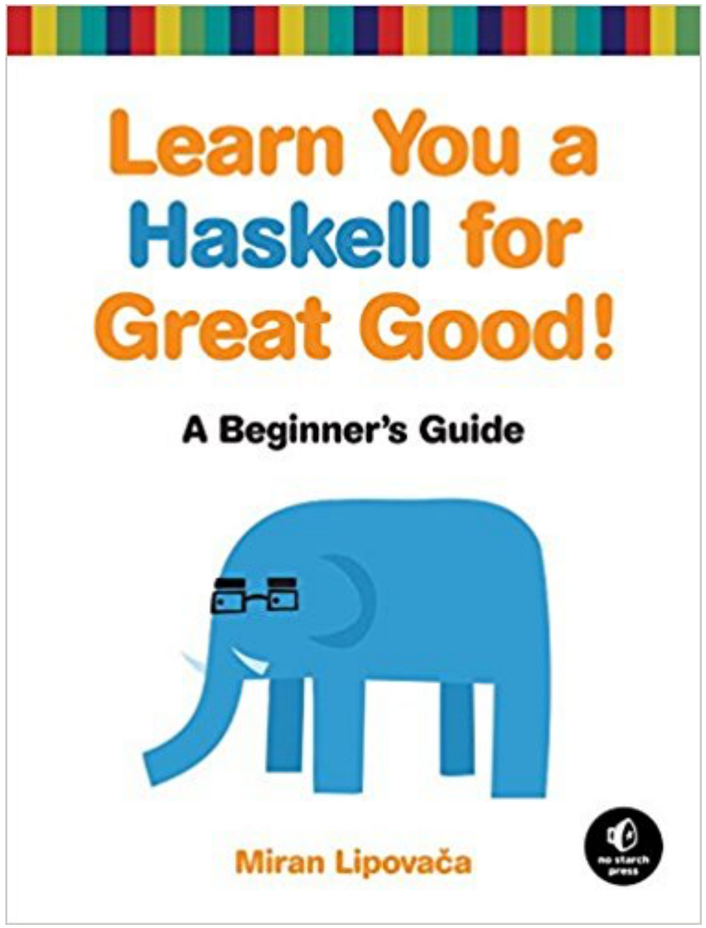
I found the data online at https://eclipse.gsfc.nasa.gov/SEpath/SEpath2001/SE2017Aug21Tpath.html

I did a bit of work to make the data readable for my program (latitude and longitude for the center of the path and the northern and southern limits), and then grabbed a map.

The hardest part was translating the lat/long values onto the map, and it isn't perfect, but the math wasn't too hard.

http://stanford.edu/~cgregg/eclipse/

Haskell



Haskell is a *completely* different way of programming than you are used to (though not as different as some of the esoteric languages)

Haskell is a "purely functional" programming language, meaning that all computation is the result of evaluating mathematical functions. There is no concept of "mutable" data in Haskell, and all functions *only* depend on their arguments and no other information.

The book *Learn You a Haskell for Great Good!* is a terrific way to learn the language!

LYaHfGG

Haskell Example (from LYaHfGG)

sample.hs:

```
doubleMe x = x + x

doubleUs x y = x*2 + y*2

doubleSmallNumber x = if x > 100
                        then x
                        else x*2
```

Functions are defined simply. If statements look a little different, too.

Haskell Example (from LYaHfGG)

```
doubleMe x = x + x

doubleUs x y = x*2 + y*2

doubleSmallNumber x = if x > 100
                        then x
                        else x*2
```

Sample functions (sample.hs)

Testing the functions

```
$ ghci
GHCi, version 8.0.2: http://www.haskell.org/ghc/  :? for help
Prelude> :set prompt "ghci> "
ghci> :l sample.hs
[1 of 1] Compiling Main             ( sample.hs, interpreted )
Ok, modules loaded: Main.
ghci> doubleMe 4
8
ghci> doubleMe 8.2
16.4
ghci> doubleUs 5 4
18
ghci> doubleSmallNumber 12
24
ghci> doubleSmallNumber 120
120
ghci>
```

Lists in Haskell (from LYaHfGG)

A common task is putting two lists together. This is done by using the `++` operator.

```
ghci> [1,2,3,4] ++ [9,10,11,12]
[1,2,3,4,9,10,11,12]
ghci> "hello" ++ " " ++ "world"
"hello world"
ghci> ['w','o'] ++ ['o','t']
"woot"
```

Watch out when repeatedly using the `++` operator on long strings. When you put together two lists (even if you append a singleton list to a list, for instance: `[1,2,3] ++ [4]`), internally, Haskell has to walk through the whole list on the left side of `++`. That's not a problem when dealing with lists that aren't too big. But putting something at the end of a list that's fifty million entries long is going to take a while. However, putting something at the beginning of a list using the `:` operator (also called the cons operator) is instantaneous.

Lists in Haskell (from LYaHfGG)

A common task is putting two lists together. This is done by using the `++` operator.

```
ghci> [1,2,3,4] ++ [9,10,11,12]
[1,2,3,4,9,10,11,12]
ghci> "hello" ++ " " ++ "world"
"hello world"
ghci> ['w','o'] ++ ['o','t']
"woot"
```

Watch out when repeatedly using the `++` operator on long strings. When you put together two lists (even if you append a singleton list to a list, for instance: `[1,2,3] ++ [4]`), internally, Haskell has to walk through the whole list on the left side of `++`. That's not a problem when dealing with lists that aren't too big. But putting something at the end of a list that's fifty million entries long is going to take a while. However, putting something at the beginning of a list using the `:` operator (also called the cons operator) is instantaneous.

How do you think Haskell stores its lists?

Lists in Haskell (from LYaHfGG)

A common task is putting two lists together. This is done by using the `++` operator.

```
ghci> [1,2,3,4] ++ [9,10,11,12]
[1,2,3,4,9,10,11,12]
ghci> "hello" ++ " " ++ "world"
"hello world"
ghci> ['w','o'] ++ ['o','t']
"woot"
```

Watch out when repeatedly using the `++` operator on long strings. When you put together two lists (even if you append a singleton list to a list, for instance: `[1,2,3] ++ [4]`), internally, Haskell has to walk through the whole list on the left side of `++`. That's not a problem when dealing with lists that aren't too big. But putting something at the end of a list that's fifty million entries long is going to take a while. However, putting something at the beginning of a list using the `:` operator (also called the cons operator) is instantaneous.

How do you think Haskell stores its lists?

As linked lists!

Lists operations in Haskell (from LYaHfGG)

`head` takes a list and returns its head. The head of a list is basically its first element.

```
ghci> head [5,4,3,2,1]
5
```

`tail` takes a list and returns its tail. In other words, it chops off a list's head.

```
ghci> tail [5,4,3,2,1]
[4,3,2,1]
```

`last` takes a list and returns its last element.

```
ghci> last [5,4,3,2,1]
1
```

`init` takes a list and returns everything except its last element.

```
ghci> init [5,4,3,2,1]
[5,4,3,2]
```

Haskell FizzBuzz

```
module Main where

main :: IO ()
main = printAll $ map fizzBuzz [1..100]
       where
       printAll [] = return ()
       printAll (x:xs) = putStrLn x >> printAll xs

fizzBuzz :: Integer -> String
fizzBuzz n | n `mod` 15 == 0 = "FizzBuzz"
           | n `mod` 5  == 0 = "Fizz"
           | n `mod` 3  == 0 = "Buzz"
           | otherwise       = show n
```

Doesn't look like C++!

Haskell takes some getting used to…

COBOL

Someone asked if we could talk a bit about COBOL. I had to look up how to run COBOL programs, but it isn't too difficult:

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO-WORLD.
      *> SIMPLE HELLO WORLD PROGRAM
       PROCEDURE DIVISION.
           DISPLAY 'HELLO, WORLD!'.
           STOP RUN.
```

It is really old-school: all UPPERCASE letters

It was made to be "readable", but it reminds me of this:

https://www.youtube.com/watch?v=PT_DnxmyuhA

COBOL FizzBuzz

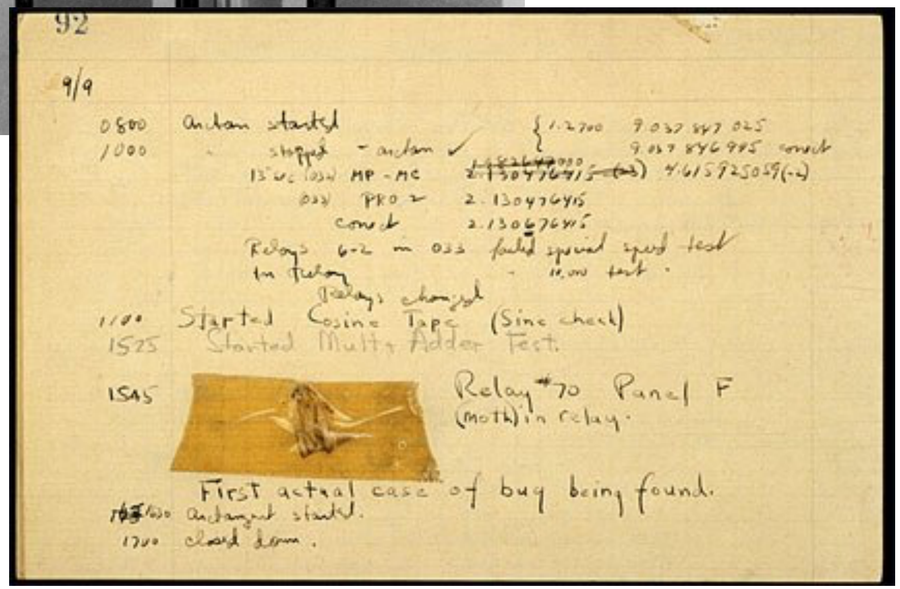
```
IDENTIFICATION DIVISION.
PROGRAM-ID. FIZZ-BUZZ.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CT PIC 999 VALUE 1.
01 FZ PIC 999 VALUE 1.
01 BZ PIC 999 VALUE 1.
PROCEDURE DIVISION.
FIZZ-BUZZ-MAIN SECTION.
    PERFORM 100 TIMES
            IF FZ = 3
                 THEN IF BZ = 5
                    THEN DISPLAY "FizzBuzz"
                    COMPUTE BZ = 0
                    ELSE DISPLAY "Fizz"
                    END-IF
                    COMPUTE FZ = 0
                 ELSE IF BZ = 5
                 THEN DISPLAY "Buzz"
                         COMPUTE BZ = 0
                 ELSE
                         DISPLAY CT
                 END-IF
          END-IF
          ADD 1 TO CT
          ADD 1 TO FZ
          ADD 1 TO BZ
    END-PERFORM
    STOP RUN.
```

Grace Hopper: Creator of COBOL

The Harvard Mark I, one of the first computers in the world

Rear Admiral, U.S. Navy

The first "bug" in the computer



- Hopper on Letterman: https://www.youtube.com/watch?v=1-vcErOPofQ

Obfucsated C

http://www.ioccc.org

```
#  include<stdio.h>//  .IOCCC                                          Fluid-  #
#  include <unistd.h>  //2012                                         _Sim!_  #
#  include<complex.h>  //||||                    ,____.               IOCCC-  #
#  define               h for(                  x=011;               2012/*  #
#  */-1>x               ++;)b[                  x]//-'               winner  #
#  define               f(p,e)                                       for(/*  #
#  */p=a;               e,p<r;                                      p+=5)//  #
#  define               z(e,i)                                      f(p,p/*  #
## */[i]=e)f(q,w=cabs   (d=*p-  *q)/2-     1)if(0  <(x=1-      w))p[i]+=w*/// ##
   double complex a [   97687]  ,*p,*q     ,*r=a,  w=0,d;    int x,y;char b/* ##
## */[6856]="\x1b[2J"  "\x1b"  "[1;1H     ", *o=  b, *t;   int main   (){/** ##
## */for(               ;0<(x=  getc (     stdin)  );)w=x  >10?32<     x?4[/* ##
## */*r++               =w,r]=  w+1,*r     =r[5]=  x==35,  r+=9:0      ,w-I/* ##
## */:(x=               w+2);;  for(;;     puts(o  ),o=b+  4){z(p      [1]*/* ##
## */9,2)               w;z(G,  3)(d*(     3-p[2]  -q[2])  *P+p[4      ]*V-/* ##
## */q[4]               *V)/p[  2];h=0     ;f(p,(  t=b+10  +(x=*p      *I)+/* ##
## */80*(               y=*p/2  ),*p+=p    [4]+=p  [3]/10  *!p[1])     )x=0/* ##
## */ <=x               &&x<79   &&0<=y&&y<23?1[1  [*t|=8   ,t]|=4,t+=80]=1/* ##
## */, *t               |=2:0;    h=" '`-.|//,\\"  "|\\_"    "\\/\x23\n"[x/** ##
## */%80-               9?x[b]      :16];;usleep(  12321)      ;}return 0;}/* ##
####                                                                    ####
########################################################################
**########################################################################*/
```

https://www.youtube.com/watch?v=QMYfkOtYYlg

Quines

A quine is a non-empty computer program which takes no input and produces a copy of its own source code as its only output.

https://en.wikipedia.org/wiki/Quine_(computing)

https://github.com/mame/quine-relay

LOLCODE FizzBuzz

```
HAI
CAN HAS STDIO?
I HAS A VAR IZ 0
IM IN YR LOOP
    UPZ VAR!!1
    IZ VAR BIGR THAN 100?
        GTFO.
    KTHX
    IZ VAR LEFTOVAR 15 LIEK 0?
        VISIBLE "FIZZBUZZ"
    KTHX
    ORLY?
    IZ VAR LEFTOVAR 5 LIEK 0?
        VISIBLE "BUZZ"
    KTHX
    ORLY?
    IZ VAR LEFTOVAR 3 LIEK 0?
        VISIBLE "FIZZ"
    KTHX
    NOWAI
        VISIBLE VAR
    KTHX
KTHX
KTHXBYE
```

References and Advanced Reading

- **References:**
  - C++ Inheritance: https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm
  - C++ Polymorphism: https://www.tutorialspoint.com/cplusplus/cpp_polymorphism.htm

- **Advanced Reading:**
  - http://stackoverflow.com/questions/5854581/polymorphism-in-c
  - https://www.codingunit.com/cplusplus-tutorial-polymorphism-and-abstract-base-class