Quiz 0 Review Session Part 0

October 14, 2013

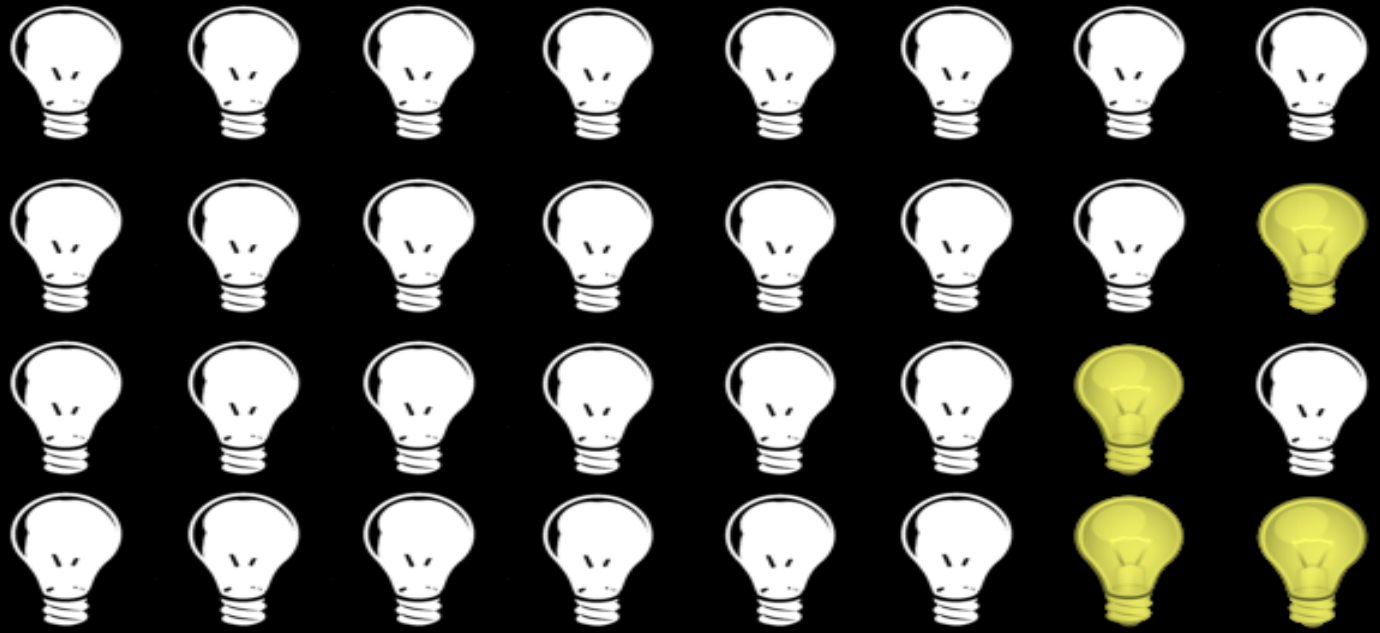
Karen Xiao

Hey! You have a quiz on Wednesday!

cs50.net/quizzes

Let’s get started!

Binary



Binary - Basics

$$\underset{2^7}{1}\ \underset{2^6}{0}\ \underset{2^5}{1}\ \underset{2^4}{0}\ \underset{2^3}{0}\ \underset{2^2}{0}\ \underset{2^1}{1}\ \underset{2^0}{1}$$

$$1*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 163$$

Binary – Binary to Decimal

$1 = 1*2^0 = 1$

$10 = 1*2^1 + 0*2^0 = 2$

$11 = 1*2^1 + 1*2^0 = 3$

$100 = 1*2^2 + 0*2^1 + 0*2^0 = 4$

$101 = 1*2^2 + 0*2^1 + 1*2^0 = 5$

Binary – Arithmetic

```
      1 1
  101011
+ 010001
--------
  111100
```

```
  1     1
  14205
+ 19418
-------
  33623
```

ASCII

- Mapping between characters and numbers
- For expressing alphabetic, numeric, and other characters in binary, the “language” that is understood by a computer

ASCII - Math

- Because characters are fundamentally just numbers, we can do math with chars!

```
int A = 65;
int B = 'A' + 1;
char C = 'D' - 1;
char D = 68;

printf("%c %c %c %c", A, B, C, D);
```

What will this print out?

ASCII - Math

- Because characters are fundamentally just numbers, we can do math with chars!

```
int A = 65;
int B = 'A' + 1;
char C = 'D' - 1;
char D = 68;

printf("%c %c %c %c", A, B, C, D);
```

What will this print out? A B C D

ASCII

- Note: ‘5’ does not equal 5
- How might we convert them?

ASCII

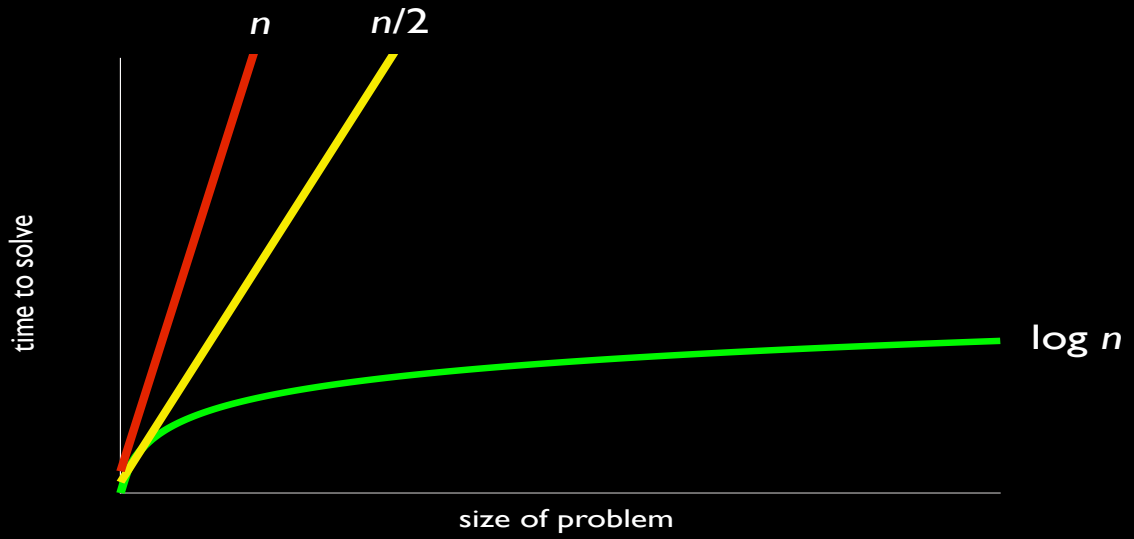
- Note: ‘5’ does not equal 5
- How might we convert them?

‘5’ – ‘0’ = 5

‘0’ + 5 = ‘5’

Algorithms

- A step-by-step set of instructions for how to perform a certain task (like a recipe?)



Pseudocode

- English-like syntax meant to represent a programming language
- Example: ask a user to guess my favorite number

```
get user's guess
if guess is correct
        tell them they are correct
else
        tell them they are not correct
```

Source Code

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("What is Karen's favorite number: ");
    int n = GetInt();

    if (n == 8)
    {
        printf("That is correct!\n");
    }
    else
    {
        printf("That is incorrect!\n");
    }
}
```

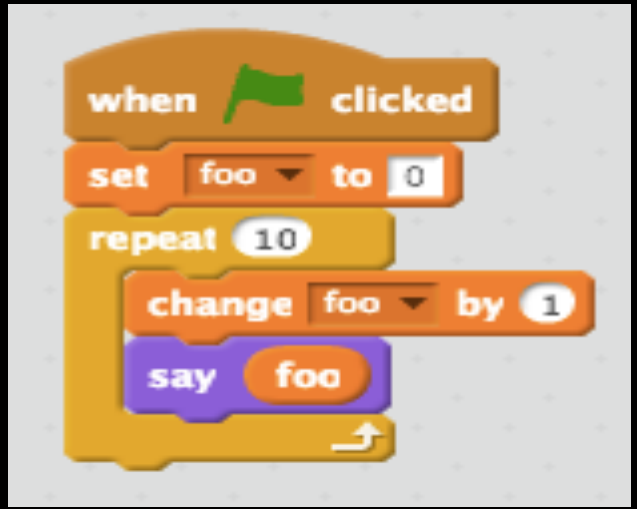
So how does your computer understand that?

Compiler

- `make` runs a compiler named `clang` for you with some command-line arguments.
- `clang` will then compile your source code to object code (0's and 1's that your computer understands)
- Source code -> Compiler -> Object code
- But more on that later…

Scratch

```
int foo = 0;
for (int i = 0; i < 10; i++)
{
    foo++;
    printf("Foo: %i\n", foo);
}
```



Let’s look at some of these building blocks that make up a program.

Boolean Expressions

Boolean expressions are those that have only two possible values: true or false, yes or no, on or off, 1 or 0.

```
bool happy = true;
if (happy)
{
    printf("smile");
}
```

Boolean Operators

&&	and
||	or
!	not
==	equal to
<=	less than or equal to
>=	greater than or equal to
<	less than
>	greater than

Conditions

Conditions are forks in the logic of a program that execute depending on whether or not certain criteria are met.

```
int x = GetInt();
if (x < 8)
{
    printf("%i is less than 8", x);
}
else if (x > 8)
{
    printf("%i is greater than 8", x);
}
else
{
    printf("%i is equal to 8", x);
}
```

Loops

```
int x;
do
{
    printf("Give me an int\n");
    x = GetInt();
}
while (x != 8);
```

Loops

- for
- while
- do while

- How do we know which one to use?

Loops

- for
  - We know how many times we want to iterate
- while
  - We need some condition to be true to keep running
- do while
  - Like while, but we want our code to run *at least once*

Loops - for

```
for (initialization; condition; update)
{
    execute this code
}
```

Loops - while

```
initialization
while (condition)
{
    execute this code
    update
}
```

Loops – do while

```
initialization
do
{
    execute this code
    update
}
while (condition);
```

Functions

- Some functions we've seen already
  - main
    - `int main(int argc, string argv[])`
  - `printf, GetInt, toupper`
    - These have been implemented for us already
- But now you can write your own!

Functions

return type

function name

parameter list

```
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

Functions – Why?

- Organization. Functions help to break up a complicated problem into more manageable subparts and help to make sure concepts flow logically into one another.

- Simplification. Smaller components are easier to design, easier to implement, and far easier to debug. Good use of functions makes code easier to read and problems easier to isolate.

- Reusability. Functions only need to be written once, and then can be used as many times as necessary, so you can avoid duplication of code.

Threads

- Threads are the concept of multiple sequences of code executing at the same time

- In Scratch, for example, multiple sprites execute scripts simultaneously
- Original example in class where we counted the number of people in the room

Events

- Events are the concept of different parts of your code “communicating” with each other

- In Scratch, this is the Broadcast/When I Receive blocks
- In Problem Set 4, `Gevent` (`waitForClick`)

Linux

- `ls`
  - stands for "list," shows the contents of the current directory
- `mkdir`
  - stands for "make directory," creates a new folder
- `cd`
  - stands for "change directory," the equivalent of double clicking on a folder
- `rm`
  - stands for "remove," deletes a file
- `rmdir`
  - stands for "remove directory," deletes a directory

Libraries

```
#include <stdio.h>
```

- what's in the stdio library?

```
#include <cs50.h>
```

- what's in the cs50 library?

Libraries

`#include <stdio.h>`

- what's in the stdio library?
  - `printf`

`#include <cs50.h>`

- what's in the cs50 library?
  - `GetInt()`, `GetString()`, etc.
  - `string`

Types

int	4 bytes
char	1 byte
float	4 bytes
double	8 bytes
long	4 bytes
long long	8 bytes
char*,int*, etc.	4 bytes

Standard Output

The printf function can take many different format codes:

- %c for char
- %i for int
- %f for float
- %lld for long long
- %s for string

A few escape sequences:

- \n for newline
- \r for carriage return (think typewriter)
- \' for single quote
- \" for double quote
- \\ for backslash
- \0 for NUL terminator