

This is Week 2

Jason Hirschhorn

Fall, 2011

Agenda

- CS50 Resources
 - Section
- Functions
 - main
 - Scope
 - Command Line Arguments
- Arrays
 - Strings
 - Multi-dimensional Arrays
- Problem Set Info
 - Magic Numbers
 - Chars to Ints
 - Caesar
 - Vigenere
- Practice Problems

CS50 Resources

- Office Hours – <https://www.cs50.net/ohs/>
- Lecture videos, slides, source code, Scribe Notes – <https://www.cs50.net/lectures/>
- Problem Set 2 Walkthrough (Sun, 7pm, NW Labs B103) – <https://www.cs50.net/psets/>
- Bulletin Board – <http://help.cs50.net>
- Me!
 - jchirschhorn@gmail.com
 - Gchat all day, every day

Our Section

- Purpose of section
 - Go over lecture material
 - Give you tools for the problem set
 - Most importantly, ensure you get what's going on
- Fun
 - Have it!
- Questions
 - Answer them!
- Support
 - I give it!
- Feedback
 - You give it!

Functions

Functions

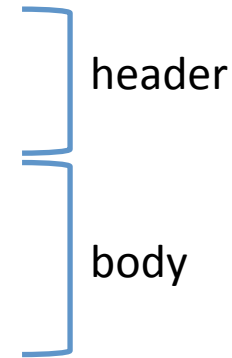
- Take things in (parameters)
- Do something to them
- Spit them out (return value)
- Why?
 - Organization, simplification, reusability



Anatomy of a Function

- Generic

```
<return type>  
<name>(<parameters>)  
{  
    <code>  
}
```



- Main

```
int  
main(void)  
{  
    <code>  
    return 0;  
}
```

Scope

- Every variable has a certain scope
 - Where the variable can be referenced
- Global vs. local
- What happens in the curly braces, stays in the curly braces!
 - Hiding

Quick Quiz

- What does the first 'a' end up as? The second? The third?

```
int a;

int
main(void)
{
    int a;
    {
        int a;
        a = 4;
    }
    a = 2;
}
```

Command Line Arguments

- How can we pass information to our program?

- From now on

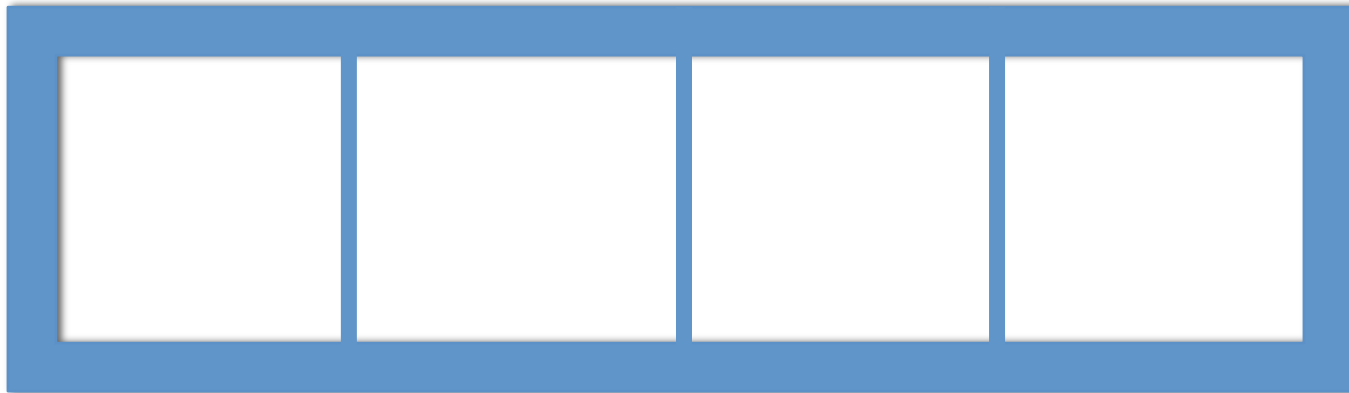
```
int  
main(int argc, char *argv[])
```

- argc
 - “Argument count”
 - # of arguments passed to the program
- argv[]
 - “Argument vector”
 - One-dimensional array of strings (each string is one of the arguments)
- ./ohai cs50 section
 - Argc = 3
 - Argv[0] = “ohai”; argv[1] = “cs50”; argv[2] = “section”

Arrays

Arrays

- Data structure to hold multiple values of the same type
- Like your mailboxes at Harvard!
 - All the same
 - Right next to each other
 - Sequentially numbered



Coding Arrays

- Create an array

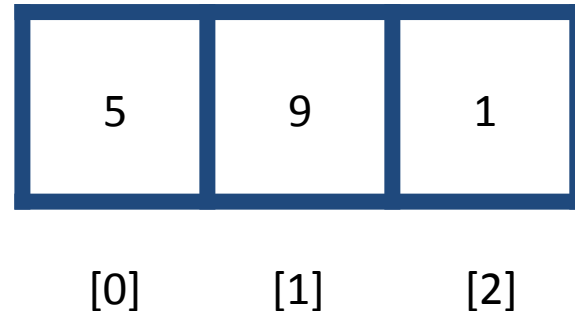
```
<data type> <name>[<size>;
```

- Access a location

```
<name>[<array index>;
```

- Example

```
int mailbox[3];  
mailbox[0] = 5;  
mailbox[1] = 9;  
mailbox[2] = 1;
```



- Alternatively

```
int mailbox[3] = {5, 9, 1};
```

- Arrays are 0-indexed

- The number in brackets refers to the offset from the first spot

Iterating Through Arrays

```
string class[3] = {"Mike", "Jane", "Charlie"};

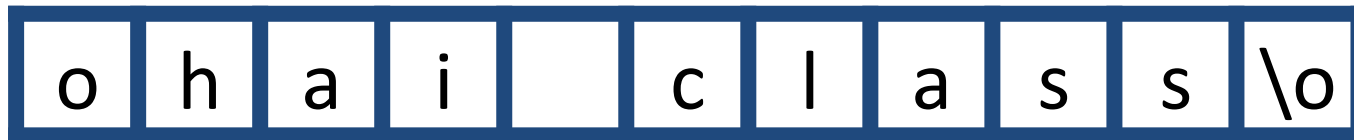
for(int i = 0; i <= 3; i++)
{
    printf("%s\n", class[i]);
}
```

Quick Quiz

- Where's the bug?
- What's `argv[argc]`?
- What's the index number of the last element in an array of size `n`?

Strings

- An array of chars!
- When iterating through a string
 - Use the null character ('\0')
 - Or, even better, strlen() (N.B. strlen doesn't count the null, so be careful when duplicating)



Quick Quiz

- Find the problem: `char foo[3] = "bar";`

Multi-dimensional Arrays

- When you want your array to have rows and columns
 - Arrays of arrays

```
char tictactoe[3][3];  
tictactoe[0][0] = 'X';  
tictactoe[0][2] = 'X';  
tictactoe[1][1] = 'O';  
tictactoe[2][0] = 'O';
```

| | | |
|---|---|---|
| X | | X |
| | O | |
| O | | |

Quick Quiz

- Given ./ohai cs50 section, what's argv[1][2]?

Problem Set Info

Magic Numbers

- Values hard coded into a program that were seemingly “pulled out of a hat”
- Bad style!

```
#define QUARTER 25
```

```
int  
main(void)  
{  
    int coins = 0;  
    int change = GetInt();  
    coins += change / QUARTER;  
}
```

Chars to Ints

- Convert between a char and its ASCII int

```
int num = 'a';
```

```
char let = 97;
```

- Cycle through the ASCII table with math!

```
char bee = 'a' + 1;
```

Quick Quiz

- What's the output: `printf("%d", num);`
- What's the output: `printf("%c", let);`

Problem Set 2

- Oldman.c
- Cryptography
 - Caesar.c
 - Vigenere.c
- Terms
 - Plaintext – the text you're going to encrypt
 - Ciphertext – the encrypted text
 - Key – used to encrypt the plaintext

Caesar Example

| | | | | | |
|---|---|---|---|---|---|
| A | B | C | D | E | F |
| G | H | I | J | K | L |
| M | N | O | P | Q | R |
| S | T | U | V | W | X |
| Y | Z | | | | |

Example

- Plaintext: ohai
- Key: 3
- Ciphertext: rkdl

Quick Quiz

- Plaintext: crazy
- Key: 5
- Ciphertext: ?

Vigenere Example

- Shift is not constant!
 - Shift each char by a letter from the key
 - Each letter from the key is likely different
- Example
 - Plaintext: I love CS50!
 - Key: ohai

| | | | | | |
|----|----|----|----|----|----|
| a | b | c | d | e | f |
| 0 | 1 | 2 | 3 | 4 | 5 |
| g | h | i | j | k | l |
| 6 | 7 | 8 | 9 | 10 | 11 |
| m | n | o | p | q | r |
| 12 | 13 | 14 | 15 | 16 | 17 |
| s | t | u | v | w | x |
| 18 | 19 | 20 | 21 | 22 | 23 |
| y | z | | | | |
| 24 | 25 | | | | |

| | | | | | | | | | | | |
|----|--|---|---|---|----|--|---|---|---|---|---|
| I | | l | o | v | e | | C | S | 5 | 0 | ! |
| o | | h | a | i | o | | h | a | | | |
| 14 | | 7 | o | 8 | 14 | | 7 | o | | | |
| W | | s | o | d | s | | J | S | 5 | 0 | ! |

Practice Problem

Fibonacci.c

- Print out as many numbers in the Fibonacci sequence as the user requests at the command line

- Example

```
./fibonacci 10
```

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

- Requirements

- User input must be one and only one positive integer
- Use a helper function that takes an integer (the user's input) and an integer array (to store the sequence)

That was Week 2

http://www.youtube.com/watch?v=5_sfnQDr1-o