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/
• 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
  
  \[
  \text{<return type> <name>(<parameters>)} \{
  \text{<code>}
  \}
  \]

- Main
  
  \[
  \text{int main(\text{void}) \{
  \text{<code> return 0;}
  \}}
  \]
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?

```c
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
  
  ```c
  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
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
  \[
  \text{<data type> <name>[<size>]};
  \]

• Access a location
  \[
  \text{<name>[<array index>]};
  \]

• Example
  \[
  \text{int mailbox[3];}
  \text{mailbox[0] = 5;}
  \text{mailbox[1] = 9;}
  \text{mailbox[2] = 1;}
  \]

• Alternatively
  \[
  \text{int mailbox[3] = \{5, 9, 1\};}
  \]

• Arrays are 0-indexed
  • The number in brackets refers to the offset from the first spot
Iterating Through Arrays


for(int i = 0; i <= 3; i++)
{
    printf(“%s
”, 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’;
```

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
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