week2

Tommy MacWilliam

tmacwilliam@cs50.net

September 19, 2011

Announcements

- Office hours! `https://www.cs50.net/ohs/`
- Walkthrough! `https://www.cs50.net/psets/`
- CS50 Help! `https://help.cs50.net`

Today

- functions
- scope
- magic
- arrays
- `main`
- crypto

Hi!

- I'm Tommy!
- Computer Science, 2013
- Gov minor (for the lulz)

I live here

I’ve worked here

charles river analytics

And here

Functions

- block of code aimed at a **single** task
- takes input $\rightarrow$ does stuff $\rightarrow$ generates output

Functions

- DRY: **D**on't **R**epeat **Y**ourself
  - functions are reusable
- break up large code chunks
  - make a big problem into a series of smaller problems
- organize your code

Functions

```
int function(int argument1, int argument)
{
    // do stuff
    // return something
}
```

Functions

- type: any variable type, like `float`, `int`, or `string`
- name: contains letters, numbers, and underscores
  - good practice: start function names with a verb
- argument list: comma-separated inputs to the function

Functions

- example time!
  - `func1.c, func2.c, func3.c`

Variable scope

- global variables: accessible by all functions
  - defined outside of `main`
- local variables: accessible by a single block
  - defined within a block, only accessible in that block

Variable scope

- what if a global and a local have the same name?
  - local variable will always be used instead
- any function can use/modify a global variable
  - this is where things can get confusing

Variable scope

- example time!
  - `scope.c`

Magic Numbers

- seemingly random constants within your program
- `http://thedailywtf.com/Articles/Magic-Number-7.aspx`

```
for (int i = 0; i < 4; i++)
{
    // next week, you will be sad because
    // you don’t know what 4 is anymore
}
```

Magic Numbers

- bad plan.
- `#define` is your friend
  - `#define CONSTANT 4`
- gives meaning to your constants
- easily changeable throughout your program

Magic Numbers

- constants created using `#define` are NOT variables
- compiler (that `make` thing) literally replaces `CONSTANT` with 4

Arrays

- list of elements of the same type
- elements accessed by their **index** (aka position)
  - index starts at 0!
- `int array[3] = {1, 2, 3};`
- `array[1] = 4;`

Arrays

- example time!
  - `array1.c, array2.c`

Multi-dimensional Arrays

- can also have arrays of arrays!
- multi-dimensional: creates a grid instead of a list
- needs multiple indices: `int grid[3][5];`
  - 3 rows, 5 columns

Multi-dimensional Arrays

```
int grid[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

Columns

		0	1	2
Rows	0	1	2	3
	1	4	5	6

Multi-dimensional Arrays

- example time!
  - `array3.c`

main

- main is a **function** that takes 2 arguments
  - `argc`: number of arguments given
  - `argv[]`: array of arguments

main

- example time!
  - `args.c`

Cryptography

- term review:
  - cleartext: the message you want to encrypt
  - ciphertext: the message after you encrypt it
  - key: what to use when encrypting the message

Caesar

- change each character by same amount
  - key is a single number

Caesar

- key = 5, message = “hello”

$$
\begin{array}{ccccc}
\mathtt{H} & \mathtt{E} & \mathtt{L} & \mathtt{L} & \mathtt{O} \\
+ & + & + & + & + \\
\mathtt{5} & \mathtt{5} & \mathtt{5} & \mathtt{5} & \mathtt{5} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\mathtt{M} & \mathtt{J} & \mathtt{Q} & \mathtt{Q} & \mathtt{T}
\end{array}
$$

Vigenere

- change each character by a different amount
  - key is a word

Vigenere

- key = “not”, message = “here”

$$\begin{array}{cccc} \texttt{H} & \texttt{E} & \texttt{R} & \texttt{E} \\ + & + & + & + \\ \texttt{N} & \texttt{O} & \texttt{T} & \texttt{N} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \texttt{U} & \texttt{S} & \texttt{K} & \texttt{R} \end{array}$$

Let’s work together

- output the first 10 even numbers
    - we need loops and functions
- calculate the Fibonacci sequence

Feedback

- how was section today?
  - `http://tommymacwilliam.com/cs50/feedback`