C: Functions and Pointers

CS 62 - Spring 2016
Michael Bannister

Assignments

- Assignment: Continuing BooOS
  - A sample design will be posted today
  - Questions for the write up will be posted today
- Lab: Basic Input/Output in C
  - Lab write up will be post today.

Example Code

- Example 1: Sums 1
  - Introduction to playing with C
- Example 2: Points 1
  - Introduction to structs

Assignment and Call-by-Value

Semantics of assignment

- The assignment operator (=) will always make a copy, e.g., the statement `x=y;` assigns a copy of `y` to `x`.
- When programming in C make sure you understand exactly what is being copied.
- Warning: assignments happen more that you would expect.

Functions are call-by-value

- The arguments passed to a function are copied
- Modifications to the arguments in the function body are not seen by the rest of the program.

Example Code

- Example 3: Points 2
  - How does passing structs to functions work or not work?

Typedefs and structs

Current

```
struct point {
   int x;
   int y;
};
void point_print(struct point p) { ... }
```

Better

```
typedef struct point {
   int x;
   int y;
} point;
void point_print(point p) { ... }
```

Function Declaration vs. Definition

Function declaration

- Specify (1) function name; (2) return type; (3) types of all arguments
- Example: `int sum(int x, int y);`
- Functions must be declared before their first use

Function definition

- Specify the exact computation of the function
- Example: `int sum(int x, int y) {return x+y;}`
- Implicitly declare a function

Example Code

- Example 4: Sums 2
  - Forward declarations
  - Recursion

Pointers!

Address-of operator

- In C we will work directly with memory
- Place a `&` in front of a variable to get its location in memory
- Example: If `int x = 4;`, then `&x` is the location in memory where `4` is stored

Pointers!

Pointer Variables

- A variable that holds a memory address, similar to Java references
- Place a in front of a variable to declare the variable as a pointer variable
- Example: If int `*p`, then p is a variable for the memory location of an int
- Example: `p = &x` is a valid assignment

Pointers!

Dereferencing a pointer

- Place a `*` in front of a point to access the value at that memory location
- Example: `*p` is the int value `4`.

Example Code

- Example 5: Pointers 1
  - Some experiments with pointers