Slides + code available at
www.github.com/pkorth/eecs280_recitation

EECS 280

Discussion 03: Jan 28, 2015

MichiganEngineering

Slides + code available at
www.github.com/pkorth/eecs280_recitation

Agenda

- Logistics
- Brief review of lecture material
  - Trees
  - Function pointers
  - Testing code
- Project 2 pitfalls
- Work on Lab 03

MichiganEngineering

Logistics

- Lab 03
  - Function pointers
  - No pre- or post-lab survey
  - Due Friday 1/30

- Project 2
  - Recursion, trees, lists
  - Due Monday 2/2

Trees

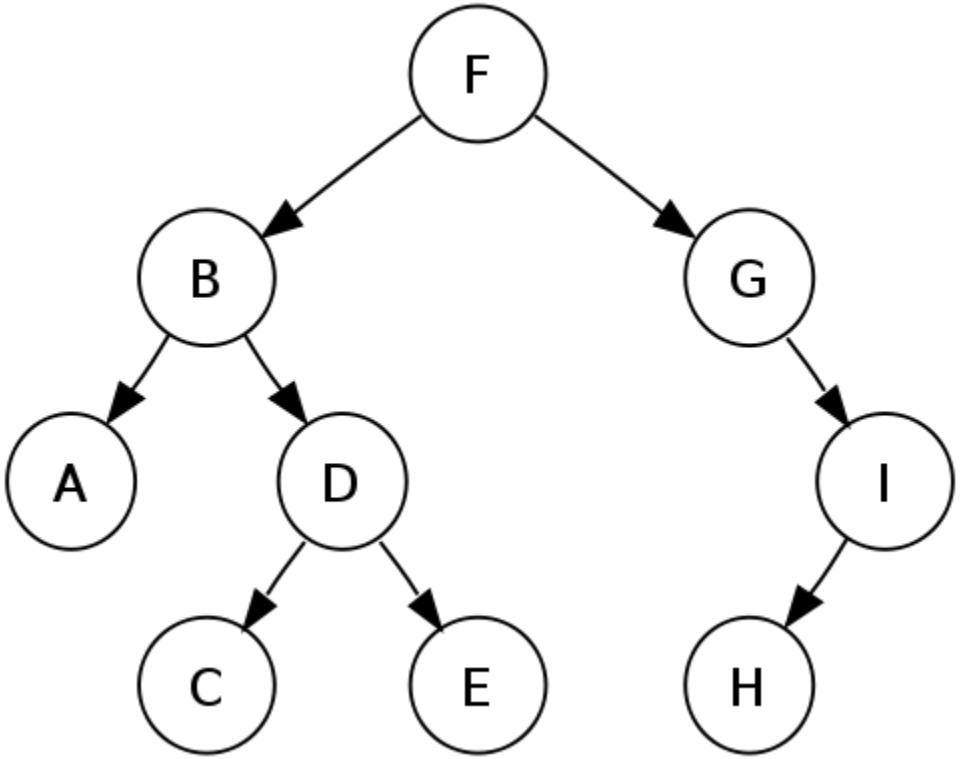

Image:
http://jeremykun.files.wordpress.com

Function pointers

Like a shortcut on your desktop, or an Internet bookmark

"Click it" and it goes to (performs) whatever it's pointing to

We can create or reassign a function pointer, or a function pointer can go out of scope, but doing so doesn't affect the underlying function

Slides + code available at
www.github.com/pkorth/eecs280_recitation

Function pointers

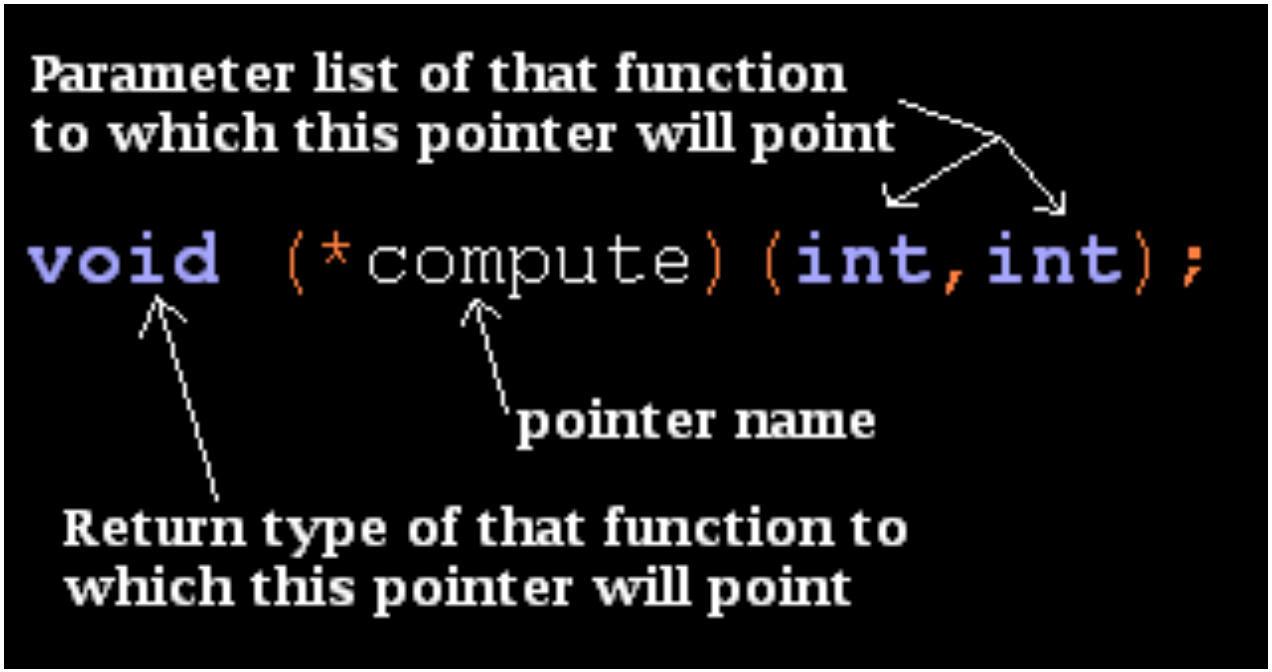

Image:
functionasargument.blogspot.com

MichiganEngineering

Function pointers

```
int *(*var1)();

bool (*var2[3])(int, double);

double *var3;

double (*(*var4)())(int);

int **var5;
```

Function pointers

Fun to try out: http://cdecl.org

(note that it won't recognize "bool" since it's in C only)

Testing code

Basic concept

test_case = abc

my_solution = MyFunction(test_case)

correct_solution = xyz

check that my_solution == correct_solution

Testing code

#include <cassert>

assert(expr)

What is the value of expr?

- True: continue running the program
- False: kill the program

“We assert that this expression must be true, otherwise fail”

Demo

Project 2 pitfalls

Calling a function and not doing anything with the result

```
1  list_t traversal(tree_t tree) {
2    // omitted code...
3
4    // INCORRECT
5
6    append(list_0, list_1);
7    append(list_1, list_2);
8    append(list_2, list_3);
9    return list_3;
10
11   // CORRECT
12
13   list_1 = append(list_0, list_1);
14   list_2 = append(list_1, list_2);
15   list_3 = append(list_2, list_3);
16   return list_3;
17 }
```

Project 2 pitfalls

Calling a function pointer instead of passing in the function pointer as an argument

```
1  void recurse(bool (*fn)(int), int i) {
2    // omitted code...
3
4    recurse(fn(5), j);     // INCORRECT
5
6    recurse(fn, j);        // CORRECT
7  }
8
9
10
11
12
13
14
15
16
17
```

Slides + code available at
www.github.com/pkorth/eecs280_recitation

Lab 03

Goal

Learn what typedef is and practice using function pointers

Tasks

1. Fill in the typedef for intCombiner
2. Implement the map() function
3. Implement the fold() function (optional)

MichiganEngineering