Introduction to Computer Science and Programming in C

Session 18: November 6, 2008

Columbia University

Announcements

- Homework 3 due next class November 11th.

Review

- big-O notation
  - Describe running time/memory
  - Ignore constant factors
- Sorting algorithms:
  - Bubble sort/Selection sort (O(n^2))
  - Merge sort (O(n log n))

Today

- Quick tidbit about structure pointers
- Program Design: Pseudocode, Headers

Structure Pointers

- Probably the most important usage of pointers
- Accessing structure fields:

```
struct business nytimes;
int size = nytimes.numEmployees;
```

-
  ```
  struct business * b_ptr = &nytimes;
  int size = (*b_ptr).numEmployees;
  ```

-
  ```
  int size = b_ptr->numEmployees;
  ```
- `(struct pointer)->(field)` is equivalent and cleaner looking

Program Design

- We have discussed in class most of the building blocks of programs
- But we still have only written small, simple programs
- Let's discuss some methods of organizing ideas so we can design larger programs

Describing Algorithms

- Up until now, I suggested describing your algorithms in English
- But English is imprecise
- We could use C instead, but C is messy

Pseudocode

- Mix of English and programming language
- Use programming constructs to keep thoughts organized: loops, conditionals, variables
- But use any syntax that is clear and consistent
- And use functions that are obvious to abstract busywork

Pseudocode example

-
  ```
  print “Enter your friends’ names:”
  while input is not “quit”
      input = keyboardInput
      add input to array Contacts

  sort Contacts
  output Contacts
  ```
- Even though this is a simple piece of code, if it were written in C, it would be much harder to understand

Pseudocode

- Forces us to be organized
- No need to look up syntax or use messy syntax
- Programmer can translate your "pseudoprogram" into any language

Header Files

- With larger programs, it’s useful to split your code into separate files
- Use headers to tie your program together.

calendar.c

struct appointment

sort()

addEvent()

cancelEvent()

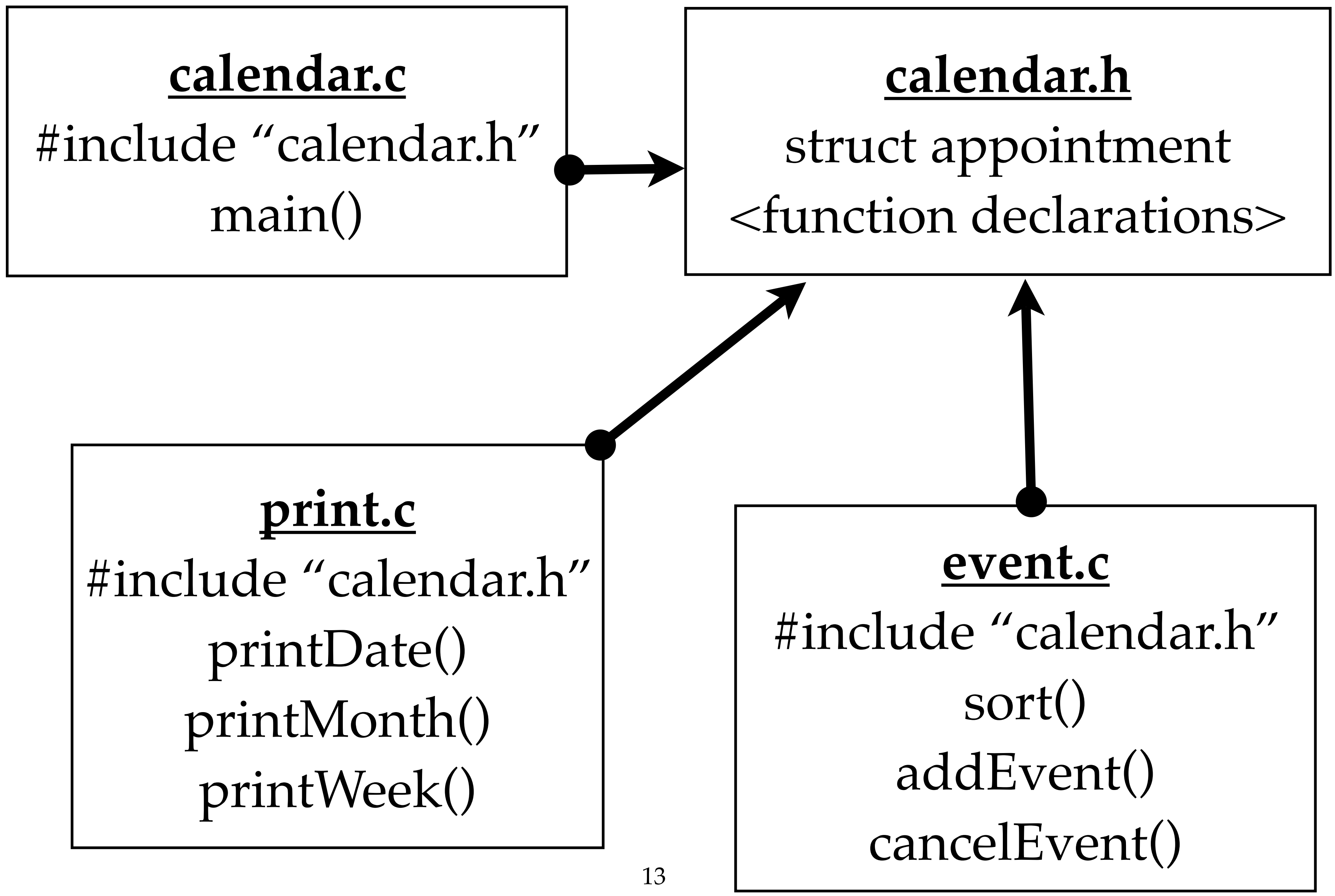
printDate()

printMonth()

printWeek()

...

main()


calendar.c
#include “calendar.h”
main()
calendar.h
struct appointment
<function declarations>
print.c
#include “calendar.h”
printDate()
printMonth()
printWeek()
event.c
#include “calendar.h”
sort()
addEvent()
cancelEvent()

extern/static Variables

- The modifier **extern** indicates that the variable is defined in a separate file.
```
extern int counter;
```

- The opposite modifier **static** indicates that the variable is only accessible to the current file.
```
static int secret_counter;
```

- With neither modifier, the variable is defined in the current file, but may be used in other files (if the other file declares with **extern**)

header Example

Reading

- For this class and next:
  Practical C Programming. Chapter 18