CSC161 2011S *Imperative Problem Solving*

Class 43: Structures

Held: Wednesday, 20 April 2011

Summary: We consider structured types in C.

Related Pages:

- EBoard.
- Lab: Structures in C.
- Reading: K&R 6.1-6.5, 6.7.
- Due: Assignment 7.

Notes:

- Our invisible visitor may be back today.
- I apologize for the confusion on the structures reading. However, there were links from the outline and syllabus and you all should know where those are. Because of the confusion, those who did not submit questions will get minuses instead of zeroes.
- Readings missing from: AA, DG, AG, JJ, JTL, EL, JY, BZ.
- No reading for Friday!
- Due to prereg and College committee work, I need folks to sign up for appointments. Times at http://www.cs.grinnell.edu/~rebelsky/Schedule/signup.2011S.txt. Signup via email.
- EC for any of the Activism Week activities, e.g., Sarah Smith on Wednesday. [academic]
- EC for Ginsburg convo, Thursday. [academic]
- EC for open discussion with Jane Ginsburg, Thursday, 9:00-10:30 in Mears Cottage. [academic]
- EC for open discussion with Jane Ginsburg, Thursday, 2:30-4:00 in Mears Cottage. [academic]
- EC for Benji's films event Saturday at 4. [peer]
- EC for Spring Bling on Friday. [peer]
- EC for Grinnell Monologues on Friday/Saturday. [peer]
- EC for ISO Cultural evening on Saturday. When else can you watch AG dance? [peer]

Overview:

- Grouping values: Arrays and Structured Types.
- Structured types in C.
- Detour: Typedef.
- Lab.

Structured Types

- Goal: Group data together because it logically falls in a group.
  - All the students in this class.
  - All the grades on HW2
  - All the information Grinnell stores about a student.
- Once grouped, you can do all sorts of cool things.
- Arrays group homogenously and index by number.
  - All the students in this class.
  - All the grades on HW2.
- Structures (“records”) in some languages, group heterogeneously and index by name.
  - All the information Grinnell stores about a student.

Structs in C

- Declared with

```
struct NAME
  {
    TYPE_0 FIELDNAME_0;
    TYPE_1 FIELDNAME_1;
    TYPE_2 FIELDNAME_2;
    ...
  };
```

- For example,

```
struct student
  {
    char *name;
    char *id;
    double gpa;
    int debt;   // In cents
    int grades[64];
  }
```

- Declare variables (and parameters) with

```
struct NAME VAR;
```

- For example,

```
struct student jordan;
```

- Basic activities:
  - “Access” a field
    - `printf ("Jordan owes approxmately $%d\n", jordan.debt/100);`
    - `jordan.debt *= 1.05; // Interest is a bear`
  - Assign one struct to another.
    - *Not in C when I learned it.*

- Declare. (Okay, you can do that with anything.)
- Get a address. (Okay, you can do that with anything.)
- *Allocate*. (You can do that with anything, but it's just been introduced.)

Detour: typedef

- Sometimes, it's nice to have a shorthand for a type name.
  - Shorthands are particularly nice for ugly types, like function pointers.
- Shorthands let you change how the program behaves by changing the type we use.
  - We might make a type-generic sort function that works with any one type we name as TYPE.
- In C, we can declare a shorthand for a type in nearly the same way we declare a variable. We just add the word `typedef` in front.
  - Variable declarations

```
int grade;
char *string;
int (*strcomparator)(char *, char*);
struct student someone;
```

  - Type declarations

```
typedef int grade;
typedef char *string;
typedef int (*strcomparator)(char *, char*);
typedef struct student someone;
```

  - And corresponding variable declarations

```
grade hw1;
string name;
strcomparator comp = strcmp;
someone jordan;
```

- As noted earlier, we can use this idea to genericize our sorting routines. (This is not a good strategy, but I thought you should see it.)

```
typedef TYPE int;
void sort (TYPE values[], int size, int compare(TYPE v0, TYPE v1));
```

Lab

- Do The lab.
- Be prepared to reflect.
