CSC161 2011S *Imperative Problem Solving*

Class 08: Types and Operators

Held: Friday, 4 February 2011

Summary: We continue our explorations of the basic design of C, focusing on types in C.

Related Pages:

- EBoard.
- Lab: C Types.
- Reading: K&R 2.1-2.6 and GNU Coding Standards, 5.1-5.4.

Notes:

- I've answered a variety of your questions on C Types and Operations. We'll look at a few.
- In case you weren't aware: There's a soda supply in the CS fridge. Students in CS classes can buy them for 50 cents. You are also welcome to make coffee.
- I understand that HW2 took longer than I'd anticipated. What was particularly problematic?
- I think I missed a few requests for the CS mailing list. If you didn't get mail about CS table last night or an REU this morning, you're not on the list. Let me know if I should add you.
- Reading for Sunday night: The rest of K&R 2, except for 2.9.
- EC for CS Table today. Informal gathering.

Overview:

- Summary of the first half of chapter 2.
- Lab.

Back to the Basics

- Let's move on to Chapter 2 of K&R. As you can tell, this chapter gives us a lot of language basics, focusing on the basic types of the language.

Variable Names

- Variable names are composed of letters, numbers, and underscores.
  - In Scheme, we got to use many other symbols.
- Depending on the C compiler you use, not all of the characters you use are significant.
  - Our compiler seems to differentiate variable names even to the 1000th character.

Declarations

- Simple form:

```
type name;
```

- In C, declarations need to come at the start of a block.
  - In a variety of C-like languages, declarations can come in many other places.

Assignment:

- Standard form

```
variable = expression;
```

- Meaning:
  - Evaluate the expression
  - Copy the value into the memory location associated with the variable

Primary Types

- C has only a few basic types: `char`, `int`, `short`, `long`, `float`, and `double`, along with a few variations of those types.
  - Signed or unsigned.
- Other types are usually represented by these types or combinations thereof.
  - For example, Boolean values are represented as integers. (0 is false, anything else is true.)
- All of these types use a fixed number of bits.
  - So the programmer is expected to understand representations
  - And we therefore need different kinds of integers.
  - You can find information on limits in `<limits.h>`
  - We'll look in detail about representations in a few days.
- Typically, C is silent about overflow.
  - If you add two numbers and the number is too large to represent, you just get some other number, rather than an error message.
- There are way too many ways to represent numeric constants
- For convenience, we'll focus on `char`, `int`, and `double`

Enums

- We can create our own "enumerated" types: Types whose values appear to be symbolic.
  - Behind the scenes, they are just integers.
- Basic syntax for describing the type

```
enum name { VAL0, VAL1, ..., VALN };
```

- Basic syntax for declaring enumerated variables

```
enum name var;
```

And More

- We’ll cover the other stuff as necessary.

Lab

- Do the lab.
- Be prepared to reflect.
