

## Class 11: Macros and More

**Held:** Wednesday, February 14, 2007

**Summary:** We start our exploration of a variety of issues with the C programming language, including the writing of tests for a common algorithm and the design and use of macros in C.

### Related Pages:

- EBoard.
- GNU Documentation: The C Preprocessor

### Overview:

- Background: Refactoring.
- A Domain: Quicksort.
- Macros: Basic Concepts.
- Enumerated Types.
- Macros: Some Hacks.

## Background: Refactoring

- Program design strategy
- Martin Fowler, coiner of the term, describes refactoring as follows

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Its heart is a series of small behavior preserving transformations. Each transformation (called a 'refactoring') does little, but a sequence of transformations can produce a significant restructuring. Since each refactoring is small, it's less likely to go wrong. The system is also kept fully working after each small refactoring, reducing the chances that a system can get seriously broken during the restructuring. (<http://www.refactoring.com/>, visited 14 February 2007)

- There are a variety of techniques for refactoring.
- From my perspective, the most common application of refactoring (and the way the term is most frequently used) is when you identify common parts of code and factor them out.
  - Good programmers factor out common parts as soon as they realize that they are about to duplicate code.
  - Copy-and-paste programming suggests an opportunity for refactoring.
- Many agile programming methodologies discourage early refactoring. That is, you write specific code first, and then only refactor when other tasks suggest that it necessary to do so.
- I've always tried to write general code, so it's a bit of a pain for me to do so.

## **A Context for Our Discussions**

- While we can talk about macros and other issues in C in the abstract, it may be helpful to look at a particular problem.
- We will look at the evolution of a Quicksort routine, similar to one I wrote for class, in the context of a growing program.

## **Generalizing the Comparison Routine**

- What if we need Quicksort to be able to sort arrays of integers in different ways (e.g., both smallest to largest and largest to smallest)?

## **Macros: Basic Concepts**

- What if experience shows that we normally sort fairly big arrays, and that the stack gets reallocated more than we'd like?
  - Yes, some people don't like calls to malloc.
  - One solution: Make the size a named constant
- What if we change the type we want to sort (say, from ints to floats)?

## **Enumerated Types**

- What if we want to declare an enumerated type of, say, students in this class?
- What changes do we have to do to make the sort routine sort them by the order in which they appear in the enumeration?
- What changes do we have to do to make the sort routine sort them "alphabetically"?

## **Macros: Some Clever Hacks**

- What if we want to sort more than one type in the same program?