Class 04: Goto Considered Harmful

Held: Monday, January 29, 2007

Summary: Today we consider the basics of structured programming and the reasons it supplanted goto-ridden programming.

Related Pages:

- EBoard.

Notes:

- Thursday at noon: Summer research opportunities.
- Questions on HW2?
- Friday’s reading is the same as Wednesday’s, so no readings distributed today.

Overview:

- Context.
- Dijkstra’s Famous Letter.
- Permutation with Repetitions.

The Context of GTSCH


- Preparation: What did you learn about context from The Story of Mel?
- Late 1960’s; about a decade after the first major high-level programming languages.
- Why did languages include go to statements?
  - Part of the underlying Von Neumann architecture.
  - At least initially, seemed like the obvious control structure. (Non-recursive procedure call also available.)
- If control was based on go to statements, why were these languages “closer to the problem”?
  - Programmers didn’t have to lay out memory for values and variables.
  - Programmers didn’t have to decompose formulae into individual operations.
  - Programmers didn’t have to worry about temporaries in such decompositions.
  - Mnemonics for many key operations.
  - Programmers didn’t have to worry about data representation.
- But people were seeing problems writing larger programs
  - As the previous reading tells us, many proto-software-engineers wanted more abstraction
  - A large community also wanted to be able to formally verify programs.
- The result: New control structures
  - Conditional of various forms (including case)
  - Recursive functions
  - Loops
- Key idea of most of these control structures (usually implicit; eventually explicit): One entry point, one exit point.
- Dijkstra, a few years earlier, had been working on parallel programming.
- Dijkstra, in some sense, was serving as (unassigned) spokesperson for a much larger group.
- The original title of the article was “A Case Against Go to”

The Letter: Go To Statement Considered Harmful

- Thesis?
- Structure of argument?
- What does he mean by “a (mixed) sequence of textual and/or dynamic indices”?
- What does he mean by “coordinate”?
- Why would one care about indices or coordinates?
- What does he mean by the “number of people in the room” example?
- Possible counter arguments?

Algorithm 306: The Interface

- What is the purpose of this algorithm?
- What distinguishes it from other algorithms for computing permutations?
- Are there other goals of the algorithm?
- Does it meet those goals?

Algorithm 306: The Implementation

- What are the arguments, the local variables, the “own” variables?
- What general purposes do they seem to serve?
- What’s going on at L2?
- Why isn’t the sequence of L4 and L5 represented as a loop?

\[
\text{L4:} \quad \text{if } a[i] = p \text{ then go to } L2; \\
\quad \text{if } a[i] < p \text{ then } r := i; \\
\text{L5:} \quad i := i - 1; \\
\quad \text{go to } L4; \\
\text{L2:} \ldots
\]

- You will continue to explore these questions in HW2.