Scott's Eight Principal Categories for Control Flow Mechanisms - 1

(from Scott, "Programming Language Pragmatics", 3rd Edition, Chapter 6, pp. 219-220)

1. Sequencing
2. Selection
3. Iteration
4. Procedural abstraction
5. Recursion
6. Concurrency
7. Exception handling and speculation
8. Nondeterminacy

Food for Thought: - 2

(from Scott, "Programming Language Pragmatics", 3rd Edition, Chapter 6, p. 220)

"Though the syntactic and semantic details vary from language to language, these eight principal categories cover all of the control-flow constructs and mechanisms found in most programming languages.

A programming who thinks in terms of these categories, rather than the syntax of some particular language, will find it easy to:

- learn new languages,
- evaluate the tradeoffs among languages, and
- design and reason about algorithms in a language-independent way."

Control flow in Assembly - 3

- often includes a JUMP command of some sort
- often includes one or more compare-then-jump commands
- from these, you can build many logical control structures (some of which will even be correct for what you want to do)

Control flow in early FORTRAN - 4

- biggest goal: designing a compiler that could produce efficient programs
- its control structures largely mirror those of the hardware they were already used to using: the IBM 704
- based on the IBM 704 branch instructions
- (such machine dependence is a characteristic of first-generation languages)

Early FORTRAN IF-statements - 5

arithmetic IF statement

```
IF (e) n1, n2, n3
```

the semantics/meaning of this statement:

- evaluate the expression e;
- if it is negative, branch to n1,
- if it is zero, branch to n2,
- if it is positive, branch to n3

"exactly the function of the 704's CAS instruction, which compares the accumulator with a value in storage and then branches to one of three locations" [MacLennan, Chapter 2]

2-way branch in early FORTRAN - 6

```
IF (condition) GOTO 100
... false case ...
GOTO 200
100 ... true case ...
200 ...


IF (.NOT. (condition)) GOTO
100
... true case ...
GOTO 200
100 ... false case ...
200 ...
```

Early FORTRAN's computed GOTO - 7

```
GOTO (10, 20, 30, 40), I
10 ... handle case 1 ...

GOTO 100

20 ... handle case 2 ...

GOTO 100

30 ... handle case 3 ...

GOTO 100

40 ... handle case 4 ...

GOTO 100

100 ...
```

More Early FORTRAN - 8

leading decision loop:

```
100 IF (loop done) GOTO 200

... body of loop ...

GOTO 100

200 ...
```

trailing decision loop:

```
100 ... body of loop ...

IF (loop not done) GOTO 100
```

definite iteration loop:

```
DO 100 I = 1, N

A(I) = A(I) * 2

100 CONTINUE
```

the dangling else problem - 9

```
if B then if C then S else T
```

- when is T done?
- Algol-60's solution - don't ALLOW an IF statement as the only statement in an IF-part!
- Pascal's/C++'s solution - pair any else with the nearest unmatched IF
- Ada's solution - have EXPLICIT ends for its IF statement (end if)