Control Structures

- A control structure is a control statement and the collection of statements whose execution it controls.
- Common controls:
  - selection
  - iteration
  - branching

Selection Statements

- Two-way selection is commonly implemented as “if-then-else”
- The control expression is usually a Boolean expression
- C, C++, Python allow arithmetic expressions
- Some require single statements in clauses (but can be blocks). Others have ways to delimint clauses (e.g. END, indentation)
- Without explicit END, else is matched with closest unmatched if.
- In functional languages, selection is an expression

Multiple-Selection Statements

- Allows one of any number of statement groups to be selected.
- Typically as “switch” or “case”
- Issues:
  - What expressions are allowed for controlling the selection?
  - How are the segments specificied?
  - Is execution flow restricted to only one segment?
  - How are case values specified?
  - Unrepresented selector values?

C/C++ switch statement

- Only discrete types (e.g. integer, character, enum) can be used in the selection expression
- Case values are compile time constants
- Allows fall through to other cases without a `break`
- Optional default case

Multiple-Selection Statements

- Other types may be allowed in selection expressions. e.g. C# allows strings
- Ruby allows any Boolean expressions
- One way to implement multiple-selection would be to use linear search through the cases. Inefficient if there are many cases
- Compiler can build a hash table of addresses to jump to
- If ranges in case values are allowed, binary search is needed
- C/C++: tables are easy to build

Multiple-Selection Statements

- Sometimes we need to use "if-then-else if then else if ..." for multiple selection
- Some languages have special words for "else-if". e.g. `elif`
- Lisp-type languages have a special form called `cond`

Iterative Statements

- Allows statements to be executed 0, 1, or more times
- Often called a loop
- There is a loop body, and a test to determine if another iteration should be performed
- The test can be a pretest or posttest

Counter-controlled Loops

- A loop variable with initial and terminal values is used to control the loop execution
- There may be a step size allowed
- Issues:
  - type and scope of loop variable
  - is loop variable read-only inside the loop body
  - loop parameters evaluated once or every iteration

C-based for loops

- Three parts: initialization, test, and increment
- Local variable can be defined in initialization
- Test evaluated each time
- No strict requirement about test or increment, any of the parts can be empty
- Used for counting, but not necessarily
- Loop variable may be modified

Python for loops

- Loop variable is assigned a value in some object, usually a range: e.g. `for x in [1,2,3,4] :`
- There is an optional else clause after termination
- `range` can be used to construct ranges

Functional Languages

- Pure functional languages have no counter and cannot implement counter-controlled loops iteratively
- Recursion is used

Logically Controlled Loops

- Whether an iteration is performed is determined by the value of some Boolean test
- Test can be done before the loop iteration (pretest) or after the loop iteration (posttest)
- while loops, do-while, repeat-until, etc.
- Posttest loops: always execute at least once
- Many languages allow for user-located loop control: break, continue, labelled break.

Iteration Based on Data Structures

- Loop over elements in some data structure
- Sometimes called “range based for loops”
- Available in Java, C++, and many newer languages (foreach)

Unconditional Branching

- Implemented by goto
- Allows control to be transferred to a specified location
- Most languages have some restrictions on possible transfer locations
- Reduces readability
- Often used for handling errors or other unusual conditions

Guarded Commands

- A guarded command is a set of Boolean expressions and associated statements to execute
- For each Boolean expression that evaluates to true, one of the associated statements is chosen to execute nondeterministically
- If none of the Boolean expression evaluates to true, a run-time error occurs
- Useful to simplify program correctness proofs, but not so useful in practical languages