Topics

- Statement-Level Control Structures
- Selection Statements
  - Two-way
  - Multiple-way

Levels of Control Flow

- Within expressions
- Among program units
- Among program statements

Control Statements

- A single control statement (a selectable goto) is minimally sufficient
- The unconditional branch statement is nonessential
- It was proven that all algorithms represented by flowcharts can be coded with only two-way selection and pretest logical loops
- Enhanced readability and writability by including more control statements.

Control Structure

- A *control structure* is a control statement and the statements whose execution it controls
- Design question
  - Should a control structure have multiple entries?

- Multiple entries add little to the flexibility of a control statement, relative to the decrease in readability caused by the increased complexity

Selection Statements

- A *selection statement* provides the means of choosing between two or more paths of execution
- Two general categories:
  - Two-way selectors
  - Multiple-way selectors

Two-Way Selection Statements

- General form:

```
if control_expression
    then clause
    else clause
```

- Design Issues:
  - What is the form and type of the control expression?
  - How are the **then** and **else** clauses specified?
  - How should the meaning of nested selectors be specified?

The Control Expression

- If the **then** reserved word or some other syntactic marker is not used to introduce the then clause, the control expression is placed in parentheses
- In C89, C99, Python, and C++, the control expression can be arithmetic
- In most other languages, the control expression must be Boolean

Clause Form

- In many contemporary languages, the then and else clauses can be single statements or compound statements
- In Perl, all clauses must be delimited by braces (they must be compound)
- In Python and Ruby, clauses are statement sequences
- Python uses indentation to define clauses

```
if x > y :
  x = y
  print "x was greater than y"
```

Nesting Selectors

- Java example

```
    if (sum == 0)
     if (count == 0)
        result = 0;
  else result = 1;
```

- Which `if` gets the `else`?
- Java's static semantics rule: `else` matches with the nearest previous `if`

Nesting Selectors (continued)

- To force an alternative semantics, compound statements may be used:

```
if (sum == 0) {
 if (count == 0)
     result = 0;
}
else result = 1;
```

- The above solution is used in C, C++, and C# too

Nesting Selectors (perl)

```
if (sum == 0) {
    if (count == 0) {
        result = 0;
    }
}
else {
    result = 1;
}
```

```
if (sum == 0) {
    if (count == 0) {
        result = 0;
    }
    else {
        result = 1;
    }
}
```

Nesting Selectors (Ruby)

```
if sum == 0 then
    if count == 0 then
        result = 0
    else
        result = 1
    end
end
```

```
if sum == 0 then
    if count == 0 then
        result = 0
    end
else
    result = 1
End
```

Nesting Selectors (Python)

```
if sum == 0 :
    if count == 0 :
        result = 0
    else :
        result = 1
```

```
if sum == 0 :
    if count == 0 :
        result = 0
else :
    result = 1
```

Selector Expressions

- In ML, F#, and Lisp, the selector is an expression; in F#:

```
let y =
  if x > 0 then x
  else 2 * x
```

  - If the `if` expression returns a value, there must be an else clause (the expression could produce a unit type, which has no value). The types of the values returned by then and else clauses must be the same.

Multiple-Way Selection Statements

- Allow the selection of one of any number of statements or statement groups
- Design Issues:
  1. What is the form and type of the control expression?
  2. How are the selectable segments specified?
  3. Is execution flow through the structure restricted to include just a single selectable segment?
  4. How are case values specified?
  5. What is done about unrepresented expression values?

Multiple-Way Selection: Examples

- C, C++, Java, and JavaScript

```
switch (expression) {
   case const_expr$_1$: stmt$_1$;
   …
   case const_expr$_n$: stmt$_n$;
   [default: stmt$_{n+1}$]
}
```

Multiple-Way Selection: Examples

- Design choices for C’s **`switch`** statement

1. Control expression can be only an integer type
2. Selectable segments can be statement sequences, or compound statements
3. Any number of segments can be executed in one execution of the construct (*there is no implicit branch at the end of selectable segments*)
4. **`default`** clause is for unrepresented values (if there is no **`default`**, the whole statement does nothing)

Multiple-Way Selection: Examples

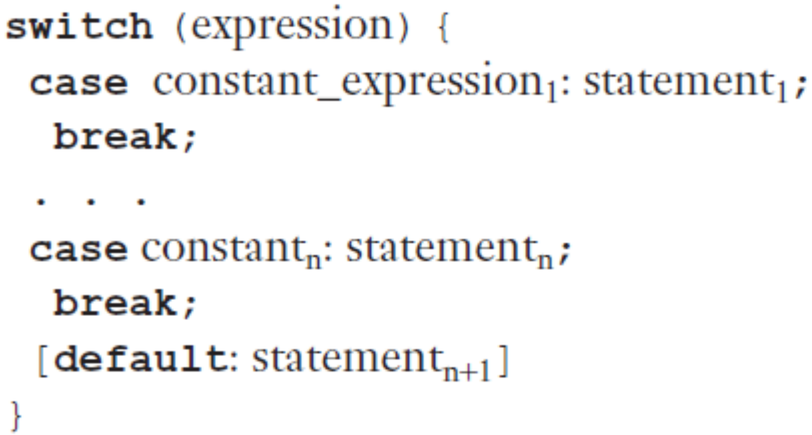
- C#
  - Differs from C in that it has a static semantics rule that disallows the implicit execution of more than one segment
  - Each selectable segment must end with an unconditional branch (`goto case`, `break`, `return`)
  - Also, in C# the control expression and the case constants can be strings

Multiple-Way Selection: Examples

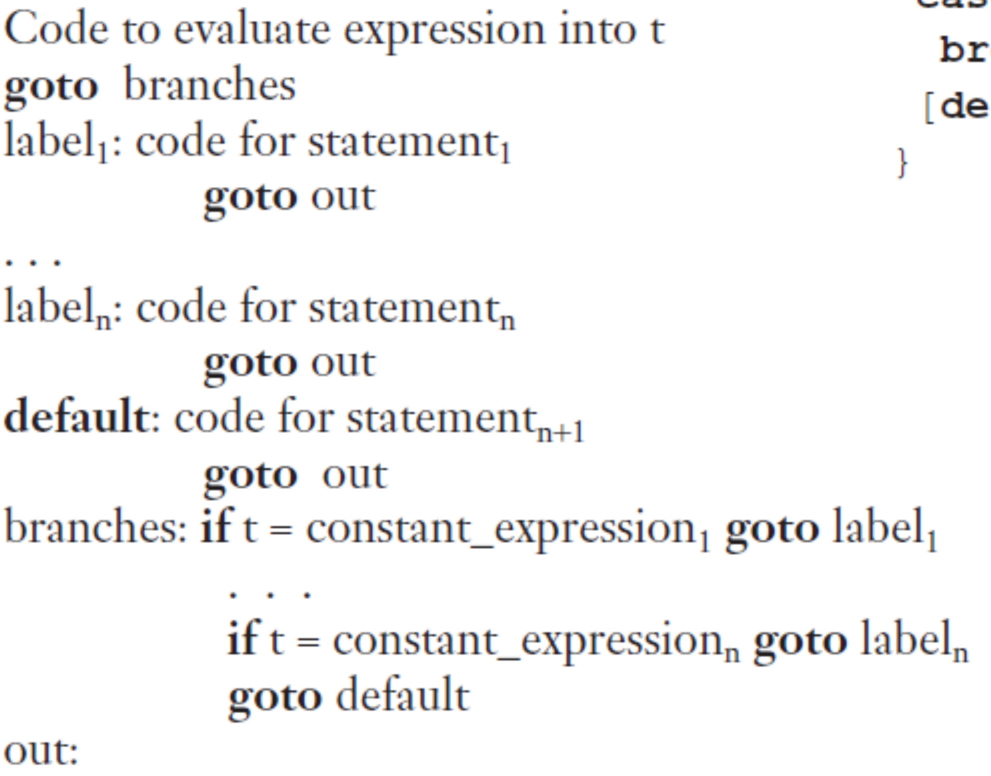
- Ruby

```
leap = case
       when year % 400 == 0 then true
       when year % 100 == 0 then false
       else year % 4 == 0
       end
```

Implementing Multiple Selectors



```
switch (expression) {
 case constant_expression$_1$: statement$_1$;
  break;
 . . .
 case constant$_n$: statement$_n$;
  break;
 [default: statement$_{n+1}$]
}
```



```
Code to evaluate expression into t
goto branches
label$_1$: code for statement$_1$
           goto out
. . .
label$_n$: code for statement$_n$
           goto out
default: code for statement$_{n+1}$
           goto out
branches: if t = constant_expression$_1$ goto label$_1$
            . . .
            if t = constant_expression$_n$ goto label$_n$
            goto default
out:
```

Implementing Multiple Selectors

- Approaches:
  - Multiple conditional branches
  - Store case values in a table and use a linear search of the table
  - When there are more than ten cases, a hash table of case values can be used
  - If the number of cases is small and more than half of the whole range of case values are represented, an array whose indices are the case values and whose values are the case labels can be used

Multiple-Way Selection Using `if`

- Multiple Selectors can appear as direct extensions to two-way selectors, using else-if clauses, for example in Python:

```
if count < 10 :
  bag1 = True
elif count < 100 :
  bag2 = True
elif count < 1000 :
  bag3 = True
```

Multiple-Way Selection Using if

- The Python example can be written as a Ruby case

```
case
  when count < 10 then bag1 = true
  when count < 100 then bag2 = true
  when count < 1000 then bag3 = true
end
```

Scheme's Multiple Selector

- General form of a call to COND:

```
(COND
  (predicate$_1$ expression$_1$)
   …
  (predicate$_n$ expression$_n$)
  [(ELSE expression$_{n+1}$)]
)
```

  - The ELSE clause is optional; ELSE is a synonym for true
  - Each predicate-expression pair is a parameter
  - Semantics: The value of the evaluation of COND is the value of the expression associated with the first predicate expression that is true