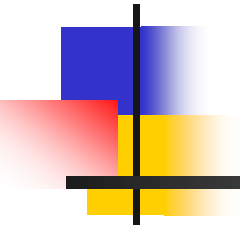
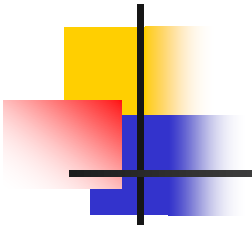
Statement-Level Control Structures



Lecture 13

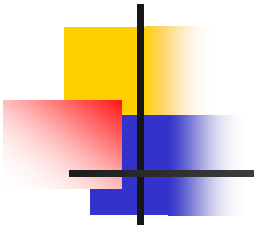
Instructor: C. Pu (Ph.D., Assistant Professor)

puc@marshall.edu



Introduction

- Computations in imperative-language programs are accomplished by
  - evaluating expressions
  - assigning the resulting values to variables
- At least two additional linguistic mechanisms are necessary to make the computations in programs flexible and powerful:
  - some means of selecting among alternative control flow paths (of statement execution)
  - some means of causing the repeated execution of statements or sequences of statements
- Statements that provide these kinds of capabilities are called ***control statements***.



Selection Statements

- A ***selection statement*** provides the means of choosing between ***two*** or ***more*** execution paths in a program.

- Such statements are fundamental and essential parts of all programming languages.

- Selection statements fall into two general categories:
  - two-way
  - n-way or multiple selection

Two-Way Selection Statements: The Control Expression

- Although the two-way selection statements of contemporary imperative languages are quite similar, there are some variations in their designs.

- The general form of a two-way selector is as follows:

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

Two-Way Selection Statements: The Control Expression

- Control expressions are specified in parentheses if the ***then*** reserved word (or some other syntactic marker) is not used to introduce the ***then*** clause.
  - In those cases where the ***then*** reserved word (or alternative marker) is used, there is less need for the parentheses, so they are often omitted, as in Ruby.

```
if number < 0 then
    print "Number is negative" E
else
    print "Number is non-negative"
```

Two-Way Selection Statements: The Control Expression

- In C89, which did not have a Boolean data type, arithmetic expressions were used as control expressions.
- This can also be done in Python, C99, and C++.
  - However, in those languages either arithmetic or Boolean expressions can be used.
- In other contemporary languages, only Boolean expressions can be used for control expressions.

Two-Way Selection Statements: Clause Form

- In many contemporary languages, the ***then*** and ***else*** clauses appear as either ***single*** statements or ***compound*** statements.

- One variation of this is Perl, in which all ***then*** and ***else*** clauses must be ***compound*** statements, even if they contain single statements.

```
if (CONDITION) {
    STATEMENT;
    ...
    STATEMENT;
}
```

Two-Way Selection Statements: Clause Form

- Many languages use braces to form compound statements, which serve as the bodies of ***then*** and ***else*** clauses.
- In Fortran 95, Ada, Python, and Ruby, the ***then*** and ***else*** clauses are statement sequences, rather than compound statements.
- The complete selection statement is terminated in these languages with a reserved word.

Two-Way Selection Statements: Clause Form

- Python uses ***indentation*** to specify ***compound*** statements.
- For example,

```
if x > y :
    x = y
    print "case 1"
```

- All statements ***equally indented*** are included in the compound statement.
- Notice that rather than ***then***, a colon is used to introduce the ***then*** clause in Python.

Two-Way Selection Statements: Nesting Selectors

- We discussed the problem of syntactic ambiguity of a straightforward grammar for a two-way selector statement.
- That ambiguous grammar was as follows:

<if_stmt> → if <logic_expr> then <stmt>
| if <logic_expr> then <stmt> else <stmt>

- The issue was that when a selection statement is nested in the ***then*** clause of a selection statement, it is not clear to which ***if*** an ***else*** clause should be associated.

Two-Way Selection Statements: Nesting Selectors

- Consider the following Java-like code: (two different interpretations)

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

Two-Way Selection Statements: Nesting Selectors

- To force the alternative semantics in Java, the inner ***if*** is put in a compound, as in

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

Two-Way Selection Statements: Nesting Selectors

- C, C++, and C# have the same problem as Java with selection statement nesting.
- Because Perl requires that all ***then*** and ***else*** clauses be compound, it does not (because brace is required).
- In Perl, the previous code would be written as

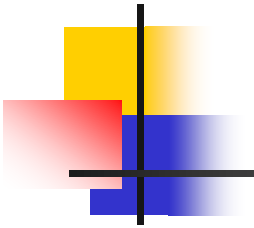
```
if (sum == 0) {
   if (count == 0) {
       result = 0;
   }
} else {
   result = 1;
}
```

Two-Way Selection Statements: Nesting Selectors

- If the alternative semantics were needed, it would be

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

Two-Way Selection Statements: Nesting Selectors



- Another way to avoid the issue of nested selection statements is to use an alternative means of forming compound statements.
- The use of a ***special word*** resolves the question of the semantics of nested selectors and also adds to the readability of the statement.
- This is the design of the selection statement in Fortran 95+, Ada, Ruby, and Lua.

Two-Way Selection Statements: Nesting Selectors

- For example, consider the following Ruby statement:

```
if a > b then
   sum = sum + a
   acount = acount + 1
else
   sum = sum + b
   bcount = bcount + 1
end
```

Two-Way Selection Statements: Nesting Selectors

- Recall that in Ruby, the ***then*** and ***else*** clauses consist of statement sequences rather than compound statements.
- The first interpretation of the selector example at the beginning of this lecture, in which the else clause is matched to the nested if, can be written in Ruby as follows:

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

Two-Way Selection Statements: Nesting Selectors

- The second interpretation of the selection statement at the beginning of this lecture, in which the else clause is matched to the outer if, can be written in Ruby as follows:

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

Two-Way Selection Statements: Nesting Selectors

- The following statement, written in Python, is semantically equivalent to the last Ruby statement above:

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

Two-Way Selection Statements: Multiple-Selection Statements

- The ***multiple-selection*** statement allows the selection of one of any number of statements or statement groups.
  - It is, therefore, a generalization of a selector.
- In fact, two-way selectors can be built with a multiple selector.
- The need to choose from among more than two control paths in a program is common.
- Although a multiple selector can be built from two-way selectors and ***gotos***, the resulting structures are cumbersome, unreliable, and difficult to write and read.
  - Therefore, the need for a special structure is clear.

Two-Way Selection Statements: Examples of Multiple Selectors

- The C multiple-selector statement, ***switch***, which is also part of C++, Java, and JavaScript, is a relatively primitive design.

- Its general form is

```
switch (expression) {
  case constant_expression1 : statement1 ;
  . . .
  case constantn : statement_n ;
  [default : statementn+1 ]
}
```

Two-Way Selection Statements: Examples of Multiple Selectors

- The ***switch*** statement does not provide implicit branches at the end of its code segments.
  - This allows control to flow through more than one selectable code segment on a single execution.
- Consider the following example:

```
index = 1;
switch (index) {
    case 1:
    case 3: System.out.println("Case 2");
    case 2:
    case 4: System.out.println("Case 4");
    default: System.out.println("Error in switch");
}
```

Two-Way Selection Statements: Examples of Multiple Selectors

- The following switch statement uses break to restrict each execution to a single selectable segment:

```
index = 1;
switch (index) {
    case 1:
    case 3: System.out.println("Case 2");
            break;
    case 2:
    case 4: System.out.println("Case 4");
            break;
    default: System.out.println("Error in switch");
}
```

Two-Way Selection Statements: Multiple Selection Using if

- In many situations, a ***switch*** or ***case*** statement is inadequate for multiple selection.
  - For example, when selections must be made on the basis of a *Boolean expression* rather than some ordinal type, nested two-way selectors can be used to simulate a multiple selector.
- To alleviate the poor readability of deeply nested two-way selectors, some languages, such as Perl and Python, have been extended specifically for this use.
- The extension allows some of the special words to be left out.
- In particular, else-if sequences are replaced with a single special word, and the closing special word on the nested if is dropped.
- The nested selector is then called an else-if clause.

Two-Way Selection Statements: Multiple Selection Using if

- Consider the following Python selector statement (note that ***else-if*** is spelled ***elif*** in Python):

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

Two-Way Selection Statements: Multiple Selection Using if

- which is equivalent to the following:

```
if count < 10 :
    bag1 = True
else :
    if count < 100 :
        bag2 = True
    else :
        if count < 1000 :
            bag3 = True
        else :
            bag4 = True
```