Other Control Structures

CSCE 110

Influenced by material developed by James Tam & Jennifer Welch
Revisiting Selection (Decision-Making) In Pascal

Selection / branching constructs (mechanisms) in Pascal
• If-then
• If-then-else
• If, else-if
• ______________
___________ Statements

An alternative to the if, else-if: at most only one of many conditions can be true, i.e. mutual exclusion

Format:

```
case (selector_expression) of
  i_1:     body;
  i_2:     body;
  ...      
  i_n:     body;
else      . . .
```

The selector expression can be a:

• __________
• __________
• __________

That evaluates to a:

• __________
• __________
• __________

The body is a ________ statement (can be a __________ statement)
Case Statement: Integer Example

Example

```pascal
case (gpa) of
  4:
    writeln('You got an A');
  3:
    writeln('You got a 'B');
  2:
    writeln('You got a C');
  1:
    writeln('You got a D');
  0:
    writeln('You got an F');
else
  writeln('GPA must be one of 4, 3, 2, 1 or 0');
end; (* case *)
```
Case Statement: Character Example

Example

```pascal
case (letter) of
  'A':
    writeln('GPA = 4');
  'B':
    writeln('GPA = 3');
  'C':
    writeln('GPA = 2');
  'D':
    writeln('GPA = 1');
  'F':
    writeln('GPA = 0');
else
  writeln('Letter grade must be one of an ''A'', ',
    '''B'', ''C'', ''D'' or ''F''');
end; (* case *)
```
Case Statement: Menu Example

while (menuChoice<>'q') and (menuChoice<>'Q') do
begin
    displayMenu;
    getMenuChoice(menuChoice);
    case (menuChoice) of
    'f', 'F':
        begin
            getNumber(num);
            answer := factorial(num);
            outputAnswer(num, answer);
        end;
    's', 'S':
        begin
            getNumber(num);
            answer := sqrt(num);
            outputAnswer(num, answer);
        end;
end; (* case *)
end
## Recap: What Decision Making Constructs Are Available In Pascal/When To Use Them

<table>
<thead>
<tr>
<th>Construct</th>
<th>When To Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If-then</strong></td>
<td>Evaluate a Boolean expression and execute some code (body) if it’s true</td>
</tr>
<tr>
<td><strong>If-then-else</strong></td>
<td>Evaluate a Boolean expression and execute some code (first body) if it’s true, execute alternate code (second body) if it’s false</td>
</tr>
<tr>
<td><strong>Multiple if’s</strong></td>
<td>Multiple Boolean expressions need to be evaluated with the answer for each expression being independent of the answers for the others (non-exclusive). Separate code (bodies) can be executed for each expression.</td>
</tr>
<tr>
<td><strong>If, else-if</strong></td>
<td>Multiple Boolean expressions need to be evaluated but zero or at most only one of them can be true (mutually exclusive). Zero bodies or exactly one body will execute.</td>
</tr>
<tr>
<td><strong>Case-of</strong></td>
<td>Similar to the ‘if, else-if” but results in smaller (cleaner) programs but only works for specific situations (Boolean expressions that involve characters or integer values only).</td>
</tr>
</tbody>
</table>
## Recap: When To Use Compound And Nested Decision Making Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>When To Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compound decision making</td>
<td>More than one Boolean expression must be evaluated before some code (body) can execute.</td>
</tr>
<tr>
<td>Nested decision making</td>
<td>The outer Boolean expression must be true before the inner expression will even be evaluated.</td>
</tr>
</tbody>
</table>
Revisiting Loops - Basic Structure Of Loops

1) ________ the control
   a) Control – typically a variable that determines whether or not the loop executes or not.

2) ________ the control against a _________

3) ________ the ____________ of the loop

4) __________________ of the control
Types Of Loops

1. Pre-test loops
   • Check the stopping condition _________ executing the body of the loop.
   • The loop executes _____ or _____ times.
   • Pascal implementation: ___________, _____

2. Post-test loops
   • Check the stopping condition _________ executing the body of the loop.
   • The loop executes _____ or _____ times.
   • Pascal implementation: ___________
Pre-Test Loops

1. Initialize loop control

2. Check if the stopping condition has been met
   a. If it’s been met then the loop ends
   b. If it hasn’t been met then proceed to the next step

3. Execute the body of the loop (the part to be repeated)

4. Update the loop control

5. Go to step 2
Pre-Test Loop: _____

Typically used when it is ______________ how many times that the loop will execute (_________ loops). Loop executes until the loop control would go _____ the __________ condition.

**Format** (counting _____):

for *initialize control* to *final value* do

body

**Format** (counting _____):

for *initialize control* downto *final value* do

body

Note: For loops are only supposed to count ___ (‘___’) or _____ (‘_______’) by ____. If the program must go up or down by other _____ then use a ________ ________ instead.

**NEVER** ______ the _____ ______ of a Pascal _____ loop in the _____ of the loop!
First For Loop Example

Example one:

begin
  var i : integer;
  var total : integer;
  total := 0;
  for i := 1 to 5 do
    begin
      total := total + i;
      writeln('i=', i, ' total=', total);
    end; (* for *)
end.
First For Loop Example

Example one:

begin
  var i : integer;
  var total : integer;
  total := 0;
  for i := 1 to 5 do
    begin
      total := total + i;
      writeln('i=', i, ' total=', total);
    end;
  end; (* for *)
end.
Second For Loop Example

Example two

begin
  var i   : integer;
  var total : integer;
  total := 0;
  for i := 5 downto 1 do
    begin
      total := total + i;
      writeln('i=', i, ' total=',total);
    end; (* for *)
  end; (* for *)
end.
Post-Test Loops

1. Initialize loop control (sometimes not needed because initialization occurs when the control is updated)

2. Execute the body of the loop (the part to be repeated)

3. Update the loop control

4. Check if the stopping condition has been met
   a. If it’s been met then the loop ends
   b. If it hasn’t been met then return to step 2.
Post Test Loops: ______________

Can be used instead of a _________ loop if you need the loop to execute the loop ____________. (Note: A _________ can also be modified so that it is ___________ to execute at least _____ by _________ the loop control to a value that will result in a _______ evaluation of the Boolean expression). Loop executes while some Boolean expression is _______, it stops when it’s _______.

Format:

repeat

body

until (Boolean expression);

Unlike the ______________,
the repeat-until loop can have __________________ in its body
without using a __________________.
program repeatUntil (output);
begin
  var i : integer;
  i := 1;
  repeat
    begin
      writeln('i = ', i);
      i := i + 1;
    end; (* loop *)
  until (i > 5);
end.
Repeat-Until: An Example (2)

program repeatUntil (output);
begin
    var i : integer;
i := 1;
    repeat
        writeln('i = ', i);
i := i + 1;
    end; (* loop *)
until (i > 5);
end.
Solving A Problem Using Loops

Problem: Write a program that will execute a game:
• The program will randomly generate a number between one and ten.
• The player will be prompted to enter their guess.
• The program will continue the game until the player indicates that they no longer want to continue.
Repeat-Until: An Example

var guess : integer;
var answer : integer;
var choice : char;

repeat
    answer := random(10) + 1;
    write('Enter your guess: ');
    readln(guess);
    if (guess = answer) then
        writeln('You guessed correctly!')
    else
        writeln('You guessed incorrectly');
    writeln('Number was ', answer, ', your guess was ', guess);
write('Play again? Enter “n” to quit or anything else to continue');
write('Choice: ');
readln(choice);
writeln;
until (choice = 'N') OR (choice = 'n');
Repeat-U ntil: Menu Example

repeat
    displayMenu;
    getMenuChoice(menuChoice);
    case (menuChoice) of
        'f', 'F':
            begin
                getNumber(num);
                answer := factorial(num);
                outputAnswer(num, answer, 'factorial');
            end;
        's', 'S':
            begin
                getNumber(num);
                answer := sqrt(num);
                outputAnswer(num, answer, 'square root');
            end;
    end (* case *)
until (menuChoice='q') or (menuChoice='Q')

What should menuChoice be initialized to?
Recap: What Looping Constructs Are Available In Pascal/When To Use Them

<table>
<thead>
<tr>
<th>Construct</th>
<th>When To Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-test loops</td>
<td>You want the stopping condition to be checked before the loop body is executed (typically used when you want a loop to execute zero or more times).</td>
</tr>
<tr>
<td>• While-do</td>
<td>• The most powerful looping construct: you can write a ‘while-do’ loop to mimic the behavior of any other type of loop. In general it should be used when you want a pre-test loop which can be used for most any arbitrary stopping condition e.g., execute the loop as long as the user doesn’t enter a negative number.</td>
</tr>
<tr>
<td>• For</td>
<td>• A ‘counting loop’: You want a simple loop to count up or down a certain number of times.</td>
</tr>
<tr>
<td>Post-test: Repeat-until</td>
<td>You want to execute the body of the loop before checking the stopping condition (typically used to ensure that the body of the loop will execute at least once).</td>
</tr>
</tbody>
</table>
Loops

One loop executes _______ of another loop(s).

Example structure:

______ loop (runs __ times)

______ loop (runs __ times)

______ of ______ loop (runs ______ times)

Example program

var i : integer;
var j : integer;
for i := 1 to 2 do
  for j := 1 to 3 do
    writeln('i=', i, ', j=', j);
writeln('All done!');
Testing Loops

Make sure that the loop executes the ______________________.

Test conditions:
  1) Loop ______________________
  2) Loop ______________________
  3) Loop ______________________
program testLoops (input, output);
begind
  var sum : integer;
  var i : integer;
  var last : integer;
  sum := 0;
i := 1;
write ('Enter the last number in the sequence to sum : ');
readln (last);
while (i <= last) do 
begin
  sum := sum + i;
  writeln('i=', i);
i := i + 1;
end;
write ('sum=', sum);
end.