Making Decisions In Pascal

In this section of notes you will learn how to have your Pascal programs choose between alternative courses of action.

High Level View Of Decision Making For The Computer

- True
  - You win!
- False
  - Game over man
**Decision-Making In Pascal**

Decisions are questions with answers that are either true or false (Boolean)

Decision making constructs (mechanisms) in Pascal
- If-then
- If-then-else
- If, else-if
- Case-of

---

**If-Then**

Decision-making: checking if a particular condition is true

Format:

```
If (operand1 relational operator operand2) then
  body;
  additional statements;
```

Example:

```
if (age >= 18) then
  writeln('You are an adult');
  writeln('Tell me more about yourself');
```

1 Operands are referred to as expressions in Leestma and Nyhoff
2 The body of the if-then is referred to as a statement in Leestma and Nyhoff
**Allowable Operands For Boolean Expressions**

If \( (\text{operand} \quad \text{relational operator} \quad \text{operand}) \) then

Operands
- integer
- real
- boolean
- char
- const

**Allowable Relational Operators For Boolean Expressions**

If (operand \quad \text{relational operator} \quad \text{operand}) then

<table>
<thead>
<tr>
<th>Pascal operator</th>
<th>Mathematical equivalent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>=</td>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>\leq</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>\geq</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>\neq</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
**If-Then (Simple Body)**

Body of if-then consists of a single statement

Format:

```plaintext
if (Boolean expression) then
  s1;
  s2;
```

Example:

```plaintext
if (x = 1) then
  writeln('Body of if');
  writeln ('After body');
```

**If-Then (Compound Body)**

Body of if-then consists of multiple statements

Format:

```plaintext
if (Boolean expression) then
  begin
    s1;
    s2;
    ...
    sn;
  end;
```

Example:

```plaintext
if (x = 1) then
  begin
    writeln('Body of if');
    writeln ('After body');
  end;
```
**If-Then (Compound Body(2))**

Example:

```pascal
if (x = 1) then
begin
  writeln('Body of if 1');
  writeln('Body of if 2');
end;
writeln('after if');
```

**If-Then-Else**

Decision-making with two conditions (true or false)

Format:

```pascal
if (operand relational operator operand) then
  body of 'if'
else
  body of 'else'
additional statements;
```

- No semi-colon (indicates end of decision making!)
- Semi-colon (decision making is complete)
If-Then-Else

Example:

if (age >= 18) then

writeln('Adult')

else

writeln('Not an adult');

writeln('Tell me more about yourself');

If-Then-Else (Simple Body)

Body of if-then-else consists of a single statement

Format:

if (Boolean expression) then

s1

else

No semi-colon (indicates end of decision-making!)

s2;

Semi-colon (this is the end of the decision-making process!)

s3;
**If-Then-Else (Simple Body(2))**

Example:

```pascal
if (x = 1) then
  writeln('body of if')
else
  writeln('body of else');
writeIn('after if-then-else');
```

**If-Then-Else (Compound Body)**

Body of if-then-else consists of multiple statements

Format:

```pascal
if (Boolean expression) then
begin
  s1;
  :
  sn;
end
else
begin
  sn+1;
  :
  sn + m;
end;
sn + m + 1;
```

- No semi-colon (marks end of decision-making!)
- Semi-colon (this is the end of the decision-making process!)
If-Then (Compound Body(2))

Example:

```pascal
if (x = 1) then
begin
  writeln('Body of if 1');
  writeln('Body of if 2');
end
else
begin
  writeln('Body of else 1');
  writeln('Body of else 2');
end;
writeln('after if-then-else');
```

Decision-Making With Multiple Expressions

Format:

```pascal
if (Boolean expression) logical operator (Boolean expression) then
body;
```

Example:

```pascal
if (x > 0) AND (y > 0) then
  writeln ('X is positive, Y is positive');
```
Decision-Making With Multiple Expressions (2)

Built-in logical operators in Pascal

AND
OR
XOR
NOT

(NAND and NOR can be constructed by combining NOT with AND & NOT with OR)

Forming Compound Boolean Expressions
With The “OR” Operator

Format:

if (Boolean expression) OR (Boolean expression) then

body;

Example:

if (gpa > 3.7) OR (yearsJobExperience > 5) then

writeln(‘You are hired’);
**Forming Compound Boolean Expressions**

**With The “AND” Operator**

Format:

if (Boolean expression) AND (Boolean expression) then

body;

Example:

if (yearsOnJob <= 2) AND (isGoofOff = True) then

writeln(‘You are fired’);

---

**Forming Compound Boolean Expressions**

**With The “XOR” Operator**

Format:

if (Boolean expression) XOR (Boolean expression) then

body;

Example:

if (takesFirstJob = true) XOR (takesSecondJob = true) then

isEmployed := true;
Forming Compound Boolean Expressions
With The “NOT” Operator

Format:

if NOT (Boolean expression) then
body;

Examples:

if NOT (x AND y) then
writeln(‘NAND’);
if NOT (x OR y) then
writeln(‘NOR’);

Order Of The Operations

<table>
<thead>
<tr>
<th>Order</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NOT</td>
</tr>
<tr>
<td>2</td>
<td>* / DIV MOD AND</td>
</tr>
<tr>
<td>3</td>
<td>+ - OR</td>
</tr>
<tr>
<td>4</td>
<td>&lt; &gt; = &lt;= &gt;= &lt;&gt;</td>
</tr>
</tbody>
</table>
Why Bracket Boolean Expressions

Compound Boolean expressions
• e.g., if \( x > 0 \) AND \( y > 0 \) then

AND has highest priority so the 0 and \( y \) becomes operands for this operation
Nested Decision Making

One decision is made inside another

Outer decisions must evaluate to true before inner decisions are even considered

Format:

\[
\text{if (Boolean expression) then}
\]

\[
\text{if (Boolean expression) then}
\]

Outer body

Inner body

Example:

\[
\text{if (num1 > 0) then}
\]

\[
\text{if (num2 > 0) then}
\]

\[
\text{writeln('Both numbers are positive')};
\]

Nested Decision Making: The Dangling Else

\[
\text{if (x > 0) then}
\]

\[
\text{if (y > 0) then}
\]

\[
\text{writeln('x is greater than zero, y is greater than zero')}
\]

\[
\text{else}
\]

\[
\text{writeln('x is greater than zero')};
\]
The Dangling Else Reformatted

if (x > 0) then
  if (y > 0) then
    writeln('x and y greater than zero')
  else
    writeln('x greater than zero');

Decision-Making With Multiple Alternatives

if-then
  Checks a condition and executes the body of code if the condition is true
if-then-else
  Checks a condition and executes one body of code if the condition is true
  and another body if the condition is false

Approaches for multiple alternatives
  Multiple if's
  Multiple else-if's
Multiple If's: Non-Exclusive Conditions

Any, all or none of the conditions may be true (independent)

Format:

```
if (Boolean expression 1) then
    body 1;
if (Boolean expression 2) then
    body 2;
:
statements after the conditions;
```

Multiple If's: Non-Exclusive Conditions (Example)

```
if (x > 0) then
    writeln('X is positive');
if (y > 0) then
    writeln('Y is positive');
if (z > 0) then
    writeln('Z is positive');
```
Multiple If's: Mutually Exclusive Conditions

At most only one of many conditions can be true.
Can be implemented through multiple if's.

Example (for full example look in Unix under
/home/231/examples/decisions/inefficientDecisionMaking.p)

```pascal
if (gpa = 4) then
  letter := 'A';
if (gpa = 3) then
  letter := 'B';
if (gpa = 2) then
  letter := 'C';
if (gpa = 1) then
  letter := 'D';
if (gpa = 0) then
  letter := 'F';
```

Multiple If, Else-If's: Mutually Exclusive Conditions

Format:

```pascal
if (Boolean expression 1) then
  body 1
else if (Boolean expression 2) then
  body 2
    :
else
  body n;
statements after the conditions;
```
**Multiple If, Else-If's: Mutually Exclusive Conditions (Example)**

```pascal
if (gpa = 4) then 
    letter := 'A'
else if (gpa = 3) then
    letter := 'B'
else if (gpa = 2) then
    letter := 'C'
else if (gpa = 1) then
    letter := 'D'
else if (gpa = 0) then
    letter := 'F'
else
    writeln('GPA must be one of 4, 3, 2, 1 or 0');
```

*Watch your semi-colons!*

**Case Statements**

An alternative to the if, else-if (at most only one of many conditions can be true)

Format (integer):

```pascal
case (expression) of
  i1:
    body;
  i2:
    body;
  :  
iN:
    body;
else
    body;
end; (* case *)
```

The expression (variable, constant, arithmetic) must evaluate to an integer.
Case Statements: Integer Example

Example (look for complete example in Unix under /home/231/examples/decisions/caseOf1.p):

```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 Statements: Characters

Format (char):

Case (expression) of
   'c_1':
      body;
   'c_2':
      body;
   :
   'c_n':
      body;
else
      body;
end; (* case *)

The expression (variable, constant, arithmetic) must evaluate to a character.

Case Statements: Character Example

Example (look for complete example in Unix under /home/231/examples/decisions/caseOf2.p):

case (letter) of
   'A':
      writeln('GPA = 4');
   'B':
      writeln('GPA = 3');
   'C':
      writeln('GPA = 2');
   'D':
      writeln('GPA = 1');
   'F':
      writeln('GPA = 0');
Case Statements: Character Example (2)

else
    writeln('Letter grade must be one of an "A", "B", "C", "D" or "F"');
end; (* case *)

Testing Decision Making Constructs

Make sure that the body of each decision making construct executes when it should.

Test:
1) Obvious true cases
2) Obvious false cases
3) Boundary cases
**Testing Decisions: An Example**

```pascal
program testDecisions (input, output);
begin
    var num : integer;
    write('Enter a value for num: '); 
    readln(num);
    if (num >= 0) then 
        writeln('Num is non-negative: ', num)
    else
        writeln('Num is negative: ', num);
end.
```

**Avoid Using Real Values When An Integer Will Do**

```pascal
program testExample;
begin
    var num : real;
    num := 1.03 - 0.42;
    if (num = 0.61) then 
        writeln('Sixty one cents')
    else
        writeln('Not sixty one cents');
end.
```
You Should Now Know

What are the four decision making constructs available in Pascal:
• If-then
• If-then-else
• If, else-if
• Case-of
• How does each one work
• When should each one be used

How to evaluate decision making constructs:
• Tracing the execution of simple decision making constructs
• Where are semi-colons needed in decision making constructs and why are they needed
• How to evaluate nested decision making constructs

You Should Now Know (2)

How the bodies of the decision making construct are defined:
• What is the body of decision making construct
• What is the difference between decision making constructs with simple bodies and those with compound bodies

What is an operand

What is a relational operator

What is a Boolean expression

How multiple expressions are evaluated and how the different logical operators work

How to test decision making constructs