Making Decisions In Pascal

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

---

If-Then

Decision-making: checking if a particular condition is true

Format:

If (operand[1]   relational operator   operand[1]) then

body;[2]   **Boolean expression**

additional statements;

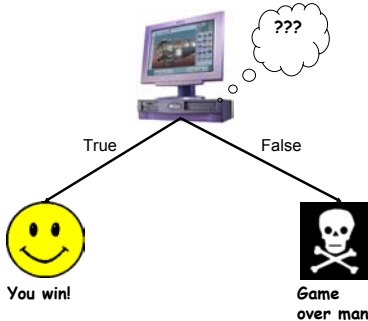   **Indicates end of decision-making**

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

---

High Level View Of Decision Making For The Computer

True   False

You win!   Game over man

Allowable Operands For Boolean Expressions

If (**operand**   relational operator   **operand**) then

Operands
- integer
- real
- boolean
- char
- const

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

Allowable Relational Operators For Boolean Expressions

If (operand   **relational operator**   operand) then

Pascal operator	Mathematical equivalent	Meaning
<	<	Less than
>	>	Greater than
=	=	Equal to
<=	≤	Less than or equal to
>=	≥	Greater than or equal to
<>	≠	Not equal to

If-Then (Simple Body)

Body of if-then consists of a single statement

Format:

if (Boolean expression) then

s1; ——— Body

s2; **Indicates end of decision-making**

Example:

if (x = 1) then

writeln('Body of if');

writeln ('After body');

If-Then-Else

Decision-making with two conditions (true or false)

Format:

if (operand relational operator operand) then

body of 'if' ——— **No semi-colon (indicates end of decision making!)**

else

body of 'else'; ——— **Semi-colon (decision making is complete)**

additional statements;

If-Then (Compound Body)

Body of if-then consists of multiple statements

Format:

if (Boolean expression) then

begin

s1;

s2;

: ——— Body

sn;

end;

sn+1; **Indicates end of decision-making**

If-Then-Else

Example:

if (age >= 18) then

writeln('Adult')

else

writeln('Not an adult');

writeln('Tell me more about yourself');

If-Then (Compound Body(2))

Example:

if (x = 1) then

begin

writeln('Body of if 1');

writeln('Body of if 2');

end;

writeln('after if');

If-Then-Else (Simple Body)

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

Format:

if (Boolean expression) then

s1 **No semi-colon (indicates end of decision-making!)**

else

s2; **Semi-colon (this is the end of the decision-making process!)**

s3;

If-Then-Else (Simple Body(2))

Example:

if (x = 1) then

    writeln('body of if')

else

    writeln('body of else')**;**

writeln('after if-then-else')**;**

Decision-Making With Multiple Expressions

Format:

if (Boolean expression) logical operator (Boolean expression) then

  body**;**

Example:

if (x > 0) AND (y > 0) then

  writeln ('X is positive, Y is positive')**;**

If-Then-Else (Compound Body)

Body of if-then-else consists of multiple statements

Format:

if (Boolean expression) then

begin

    s1**;**

      :

    sn**;**

end

else   **No semi-colon (marks end of decision-making!)**

begin

    sn+1**;**

      :

    sn + m**;**   **Semi-colon (this is the end of the decision-making**
  end**;**   **process!)**

sn + m + 1**;**

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)

If-Then (Compound Body(2))

Example:

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')**;**

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')**;**

Order Of The Operations

Order	Operator
1	NOT
2	* / DIV MOD AND
3	+ - OR
4	< > = <= >= <>

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**;**

Why Bracket Boolean Expressions

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

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')**;**

Why Bracket Boolean Expressions

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

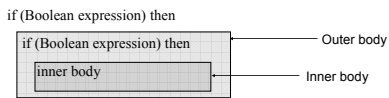AND has highest priority so the 0 and y
becomes operands for this operation

Nested Decision Making

One decision is made inside another

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

Format:

if (Boolean expression) then

Outer body

inner body
Inner body

Example:

if (num1 > 0) then

   if (num2 > 0) then

      writeln('Both numbers are positive');

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

Nested Decision Making: The Dangling Else

if (x > 0) then

if (y >0) then

writeln('x is greater than zero, y is greater than zero')

else

writeln('x is greater than zero');

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;

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');

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's);

Multiple If's: Mutually Exclusive Conditions

At most only one of many conditions can be true ←

Can be implemented through multiple if's ←

Inefficient combination!

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

```
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';
```

James Tam

Case Statements

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

Format (integer):

```
case (expression) of
    i_1:
        body;
    i_2:
        body;
        :
    i_n:
        body;
    else
        body;
end; (* case *)
```

The expression (variable, constant, arithmetic) must evaluate to an integer

James Tam

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

Format:

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

James Tam

Case Statements: Integer Example

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

```
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');
```

James Tam

Multiple If, Else-If's: Mutually Exclusive Conditions (Example)

```
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!

James Tam

Case Statements: Integer Example (2)

```
else
    writeln('GPA must be one of 4, 3, 2, 1 or 0');
end; (* case *)
```

James Tam

Case Statements: Characters

Format (char):

```
Case (expression) of
    'c₁':
        body;
    'c₂':
        body;
    :
    'cₙ':
        body;
    else
        body;
end; (* case *)
```

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

Input Testing

```
program inputChecking (input, output);
begin
    var num : integer;
    var ch  : char;
    write('Enter number and a character: ');
    read(num,ch);
    writeln('num:', num, '-ch:', ch, '-');
end.
```

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');
```

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

Case Statements: Character Example (2)

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

Testing Decisions: An Example

```
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.
```

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 perform input checking

How to test decision making constructs