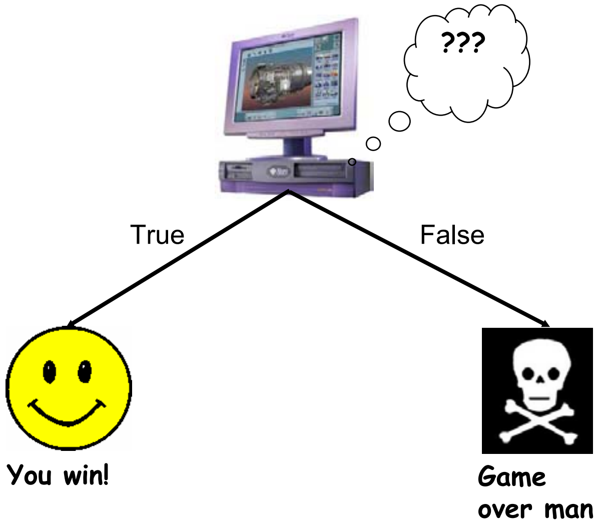
Making Decisions In Pascal

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



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 (operand[1]   relational operator   operand[1]) then
    body;[2]
additional statements;
```

Boolean expression

Indicates end of decision-making

Example:

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

Boolean expression

Indicates end of decision-making

[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 (**operand** relational operator **operand**) then

Operands

- integer
- real
- boolean
- char
- const

Allowable Relational Operators For Boolean Expressions

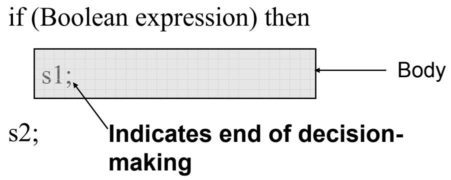
If (operand **relational operator** operand) then

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;
s2;
```

Body

Indicates end of decision-making

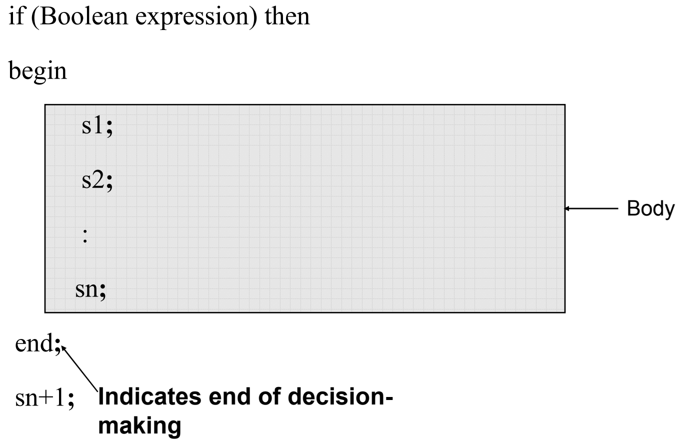
Example:

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

If-Then (Compound Body)

Body of if-then consists of multiple statements

Format:



```
if (Boolean expression) then
begin
     s1;
     s2;
     :
    sn;
 end;
 sn+1;
```

Body

Indicates end of decision-making

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

Decision-making with two conditions (true or false)

Format:

```
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
    s2;
s3;
```

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

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

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');
```

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;
```

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');
```

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');
```

Decision-Making With Multiple Expressions (2)

Built-in logical operators in Pascal

OR

AND

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

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

Why Bracket Boolean Expressions

Compound Boolean expressions

- e.g., if x > 0 AND y > 0 then

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’ become operands for this operation

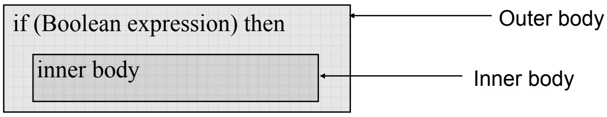
Nested Decision Making

One decision is made inside another

Outer decisions must evaluate to true before inner decisions are even considered for evaluation.

Format:

```
if (Boolean expression) then
```



```
if (Boolean expression) then
    inner body
```

Example:

```
if (num1 > 0) then
    if (num2 > 0) then
        writeln('Both numbers are positive');
```

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’);
```

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

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';
```

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;
```

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!

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

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’);
```

Case Statements: Integer Example (2)

```
   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 *)
```

Recap: What Decision Making Constructs Are Available In Pascal/When To Use Them

Construct	When To Use
If-then	Evaluate a Boolean expression and execute some code (body) if it’s true
If-then-else	Evaluate a Boolean expression and execute some code (first body) if it’s true, execute alternate code (second body) if it’s false
Multiple if’s	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.
If, else-if	Multiple Boolean expressions need to be evaluated but zero or at most only one of them can be true (mutually exclusive). Zero or one body will execute.
Case-of	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).

Recap: When To Use Compound And Nested Decision Making Constructs

Construct	When To Use
Compound decision making	More than one Boolean expression must be evaluated before some code (body) can execute.
Nested decision making	The outer Boolean expression must be true before the inner expression will even be evaluated.

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

```
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

```
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 and use 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 and compound decision making constructs and when to use them

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