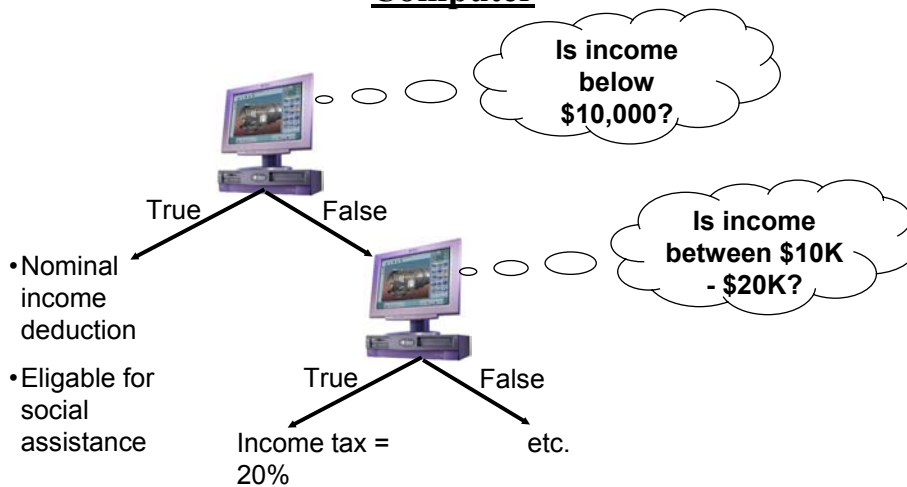


Making Decisions In Pascal

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

James Tam

High Level View Of Decision Making For The Computer



James Tam

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

James Tam

If-Then

Decision-making: checking if a particular condition is true

Format:

```
if (operand1 relational operator operand1) 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

James Tam

Allowable Operands For Boolean Expressions

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

Operands

- integer
- real
- boolean
- char
- const

James Tam

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

James Tam

If-Then (Simple Body)

Body of if-then consists of a single statement

Format:

if (Boolean expression) then

```
s1;
```

A diagram showing a grid box containing the code 's1;'. An arrow points from the text 'Body' to the right side of the box. Another arrow points from the text 'Indicates end of decision-making' to the semicolon at the end of 's1;'.

s2;

Indicates end of decision-making

Example:

if (x = 1) then

```
writeln('Body of if');
```

```
writeln ('After body');
```

James Tam

If-Then (Compound Body)

Body of if-then consists of multiple statements

Format:

if (Boolean expression) then

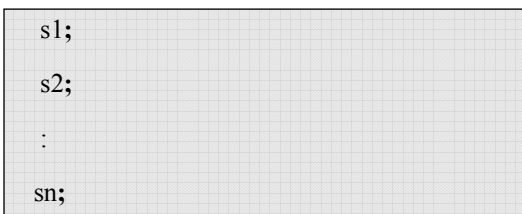
begin

```
s1;
```

```
s2;
```

```
:
```

```
sn;
```

A diagram showing a grid box containing the code 's1;', 's2;', ':', and 'sn;'. An arrow points from the text 'Body' to the right side of the box. Below the box, the code 'end;' is shown, with an arrow pointing from the text 'Indicates end of decision-making' to the semicolon at the end of 'end;'.

Indicates end of decision-making

```
end;
```

```
sn+1;
```

James Tam

If-Then (Compound Body(2))

Example:

```
if (income < 10000) then
begin
    writeln('Eligible for social assistance');
    taxCredit = 100;
end;
tax = income * TAX_RATE;
```

James Tam

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)

James Tam

If-Then-Else

Example:

```
if (age >= 18) then
    writeln('Adult')
else
    writeln('Not an adult');
writeln('Tell me more about yourself');
```

James Tam

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!)

James Tam

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');
```

James Tam

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 (not the end of decision-making process!)
begin
    sn+1;
    :
    sn + m; Semi-colon (this is the end of the decision-making process!)
end;
sn + m + 1;
```

James Tam

If-Then (Compound Body(2))

Example:

```
if (income < 10000) then
begin
    writeln('Eligible for social assistance');
    taxRate = 0.1;
end
else
begin
    writeln('Not eligible for social assistance');
    taxRate = 0.2;
end;
tax = income * taxRate;
```

James Tam

Quick Summary: If Vs. If-Else

If:

- Evaluate a Boolean expression (ask a question)
- If the expression evaluates to true then execute the 'body' of the if.
- No additional action is taken when the expression evaluates to false.
- Use when your program evaluates a Boolean expression and code will be executed only when the expression evaluates to true.

If-else:

- Evaluate a Boolean expression (ask a question)
- If the expression evaluates to true then execute the 'body' of the if.
- If the expression evaluates to false then execute the 'body' of the else.
- Use when your program evaluates a Boolean expression and different code will execute if the expression evaluates to true than if the expression evaluates to false.

James Tam

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');
```

James Tam

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)

James Tam

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');
```

James Tam

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');
```

James Tam

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;
```

James Tam

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');
```

James Tam

Order Of The Operations

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

James Tam

Why Bracket Boolean Expressions

Compound Boolean expressions

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

James Tam

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

James Tam

Quick Summary: Using Multiple Expressions

Use multiple expressions when multiple Boolean expressions must be asked and the result of each expression may have an effect on the other expressions:

AND:

- All Boolean expressions must evaluate to true before the entire expression is true.

OR:

- All Boolean expressions must evaluate to false before the entire expression is true.

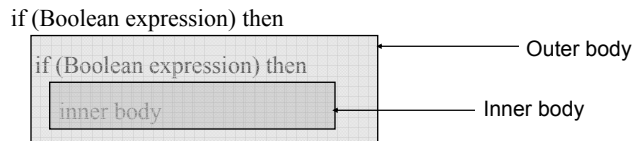
James Tam

Nested Decision Making

One decision is made inside another

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

Format:



Example:

```
if (income < 10000) then
    if (citizen = true) then
        writeln('Eligable for social assistance');
tax = income * TAX_RATE;
```

James Tam

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');
```

James Tam

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');
```

James Tam

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 (two or more) alternatives

Multiple if's

Multiple else-if's

James Tam

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;
```

James Tam

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');
```

James Tam

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/tamj/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

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

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

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

Format (integer):

```
case (expression) of
```

```
    i1:
        body;
```

```
    i2:
        body;
```

```
    :
```

```
    in:
        body;
```

```
else
```

```
    body;
```

```
end; (* case *)
```

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

James Tam

Case Statements: Integer Example

Example (look for complete example in Unix under /home/231/tamj/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

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

James Tam

Case Statements: Character Example

Example (look for complete example in Unix under
`/home/231/tamj/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');
```

James Tam

Case Statements: Character Example (2)

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

James Tam

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 bodies or exactly 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).

James Tam

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.

James Tam

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

James Tam

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.
```

James Tam

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.
```

James Tam

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

James Tam

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

James Tam