Making Decisions In Pascal

In this section of notes you will learn how to have your Pascal programs to execute alternatives

James Tam

Decision-Making In Pascal

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

Decision making statements

- If-then
- If-then-else
- case of

If-Then

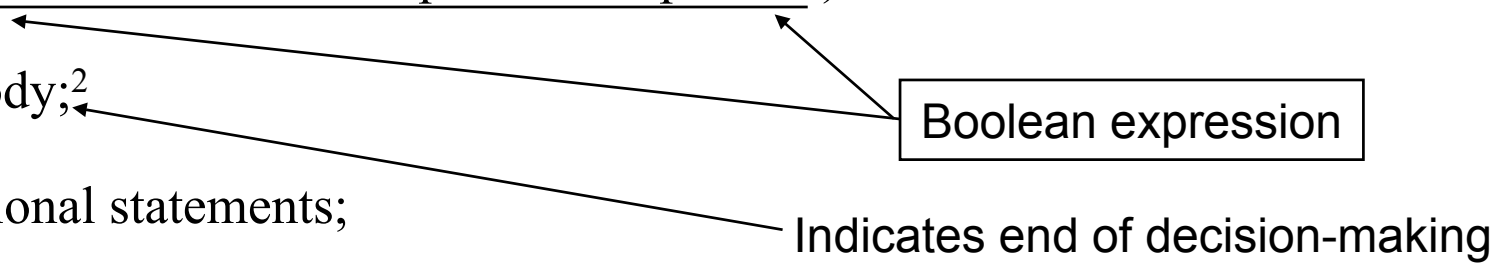
Decision-making with one condition

Format:

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

body;[2]

additional statements;



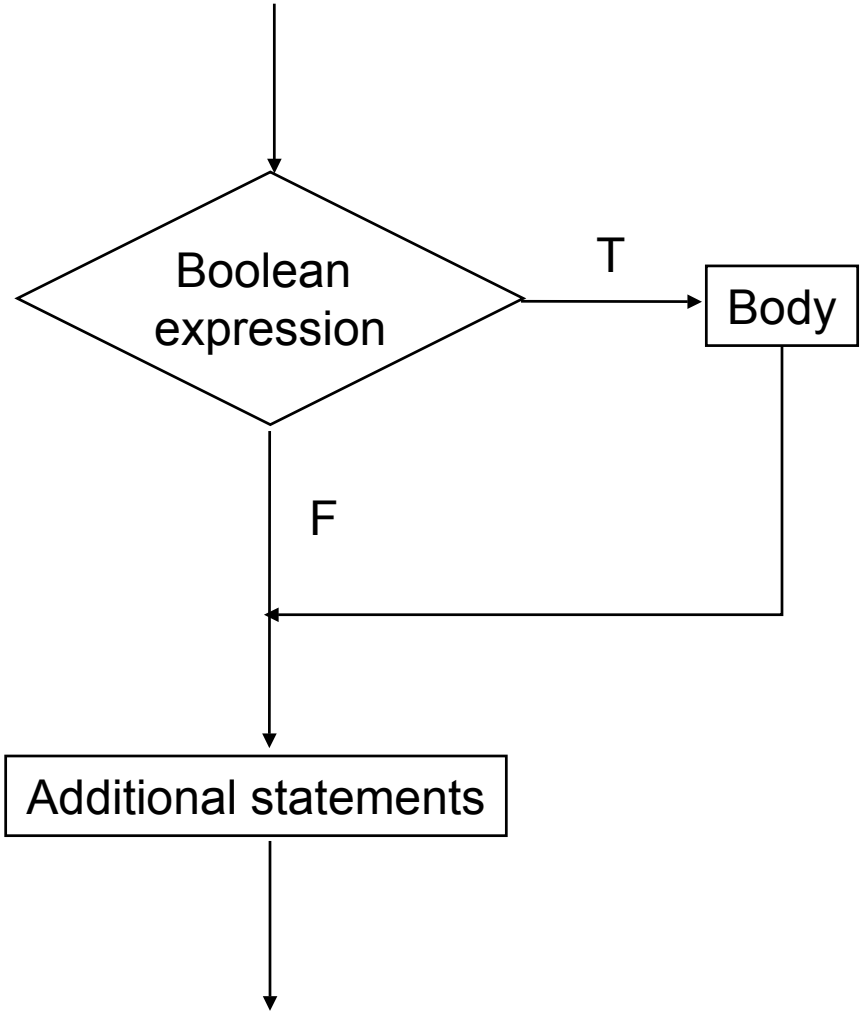
Example:

```
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 Body of the if-then is referred to as a statement in Leestma and Nyhoff

If-Then (Flowchart)

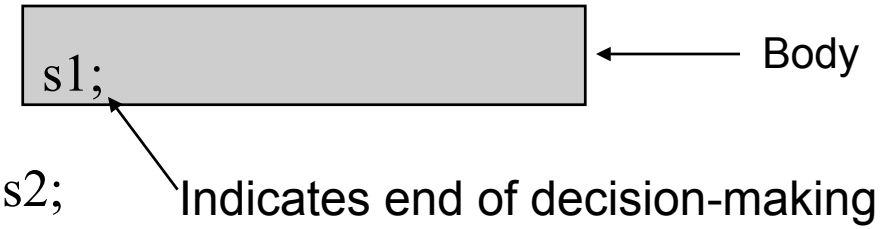


If-Then (Simple Body)

Body of if-then consists of a single statement

Format:

```
if (Boolean Expression) then
    s1;
s2;
```



Example (for full example look under /home/231/examples/decisions/simpleIfThen.p):

```
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 (for full example look under /home/231/examples/decisions/compoundIfThen.p):

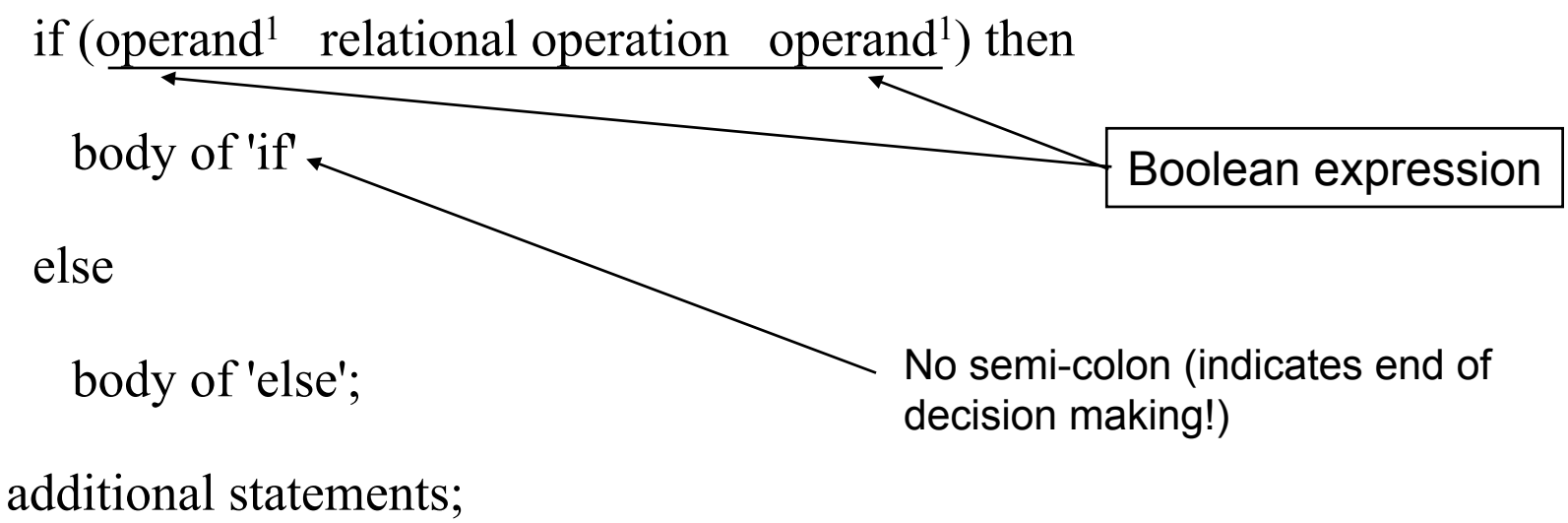
```
if (x = 1) then
    begin
        writeln('Body of if 1');
        writeln('Body of if 2');
    end;
  writeln('after if');
```

If-Then-Else

Decision-making with two conditions

One (and only one) condition will be true

Format:

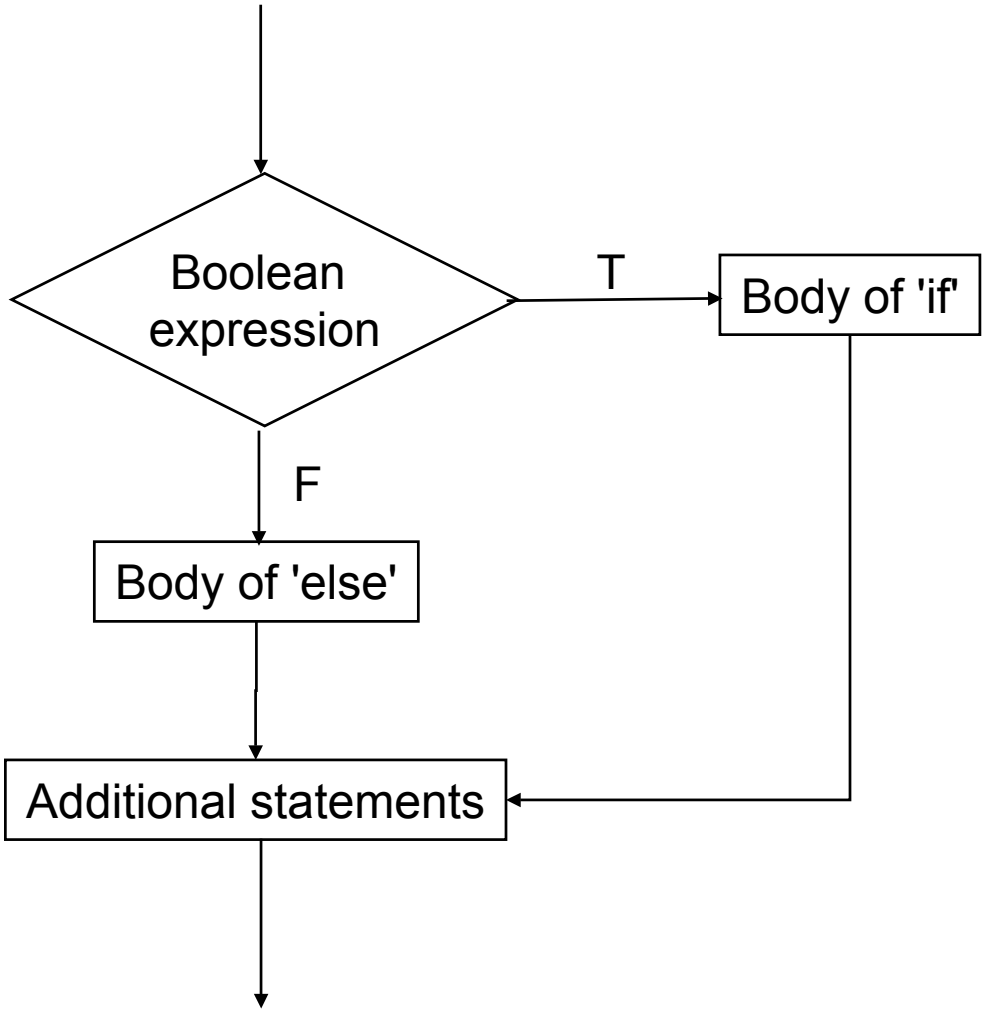


If-Then-Else

Example:

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

If-Then-Else(Flowchart)



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!)

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

Example (for full example look under /home/231/examples/decisions/simpleIfThenElse.p):

```
 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
        begin
            sn+1;
             :
            sn + m;
         end;
Sn + m + 1;
```

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

If-Then (Compound Body(2))

Example (for full example look under /home/231/examples/decisions/compoundIfThenElse.p):

```
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');
```

Allowable Operands For Boolean Expressions

If (operand relational operation operand) then

Operands

- integer
- real
- boolean
- char
- const

Allowable Operations 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

Decision-Making With Multiple Expressions

Typical format:

if (Boolean expression) Boolean operation (Boolean expression)

body

Boolean expressions

Formed from relational operations and their operands e.g., x > 5

Decision-Making With Multiple Expressions (2)

Built-in Boolean operations in Pascal

NOT

AND

OR

XOR

(NAND and NOR can be constructed via NOT, AND & OR)

Forming Compound Boolean Expressions With NOT

Format

```
if NOT (Boolean Expressions)

   body;
```

Example

```
if NOT (x AND y)

if NOT (x OR y)
```

For a complete example program look in Unix under /home/231/examples/decisions/compoundNOT.p

Forming Compound Boolean Expressions With OR

Format

```
if (Boolean Expression) OR (Boolean Expression)
   body;
```

Example

```
if (gpa > 3.7) OR (years_job_experience > 5)
   writeln('You're hired');
```

For a complete example program look in Unix under
/home/231/examples/decisions/compoundOR.p

Forming Compound Boolean Expressions With AND

Format

```
if (Boolean Expression) AND (Boolean Expression)

   body;
```

Example

```
if (years_on_job <= 2) AND (is_goof_off = True)

   writeln('You're fired');
```

For a complete example program look in Unix under /home/231/examples/decisions/compoundAND.p

Forming Compound Boolean Expressions With XOR

Format

```
if (Boolean Expression) XOR (Boolean Expression)
   body;
```

Example

```
if (takes_first_job = True) XOR (takes_second_job = True)
   is_employed := true;
```

Order Of Operation

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

Why Bracket Boolean Expressions

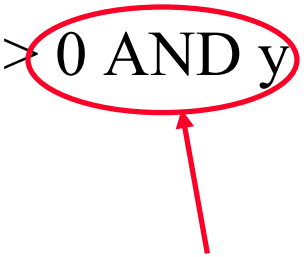
Compound Boolean Expressions

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

Why Bracket Boolean Expressions

Compound Boolean Expressions

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



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

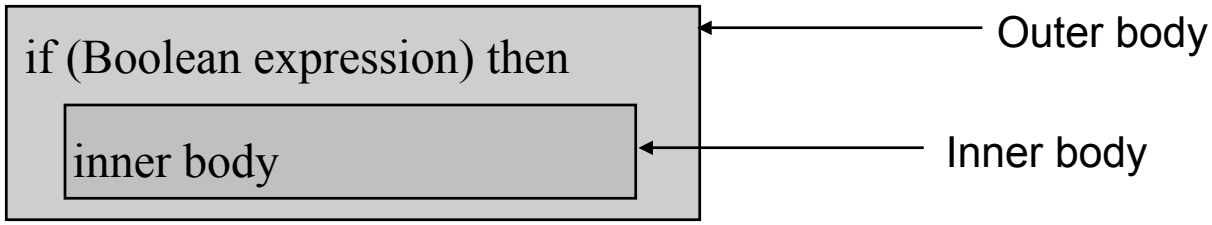
Nested Decision Making

One decision is made inside another

Outer decisions must be true before inner decisions are considered

Format

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



Example (For complete example look in Unix under /home/231/examples/decisions/nesting.p)

```
 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 one condition

if-then-else

- Checks for one of two mutually exclusive conditions

Approaches for multiple alternatives

- Multiple if's
- Multiple else-if's

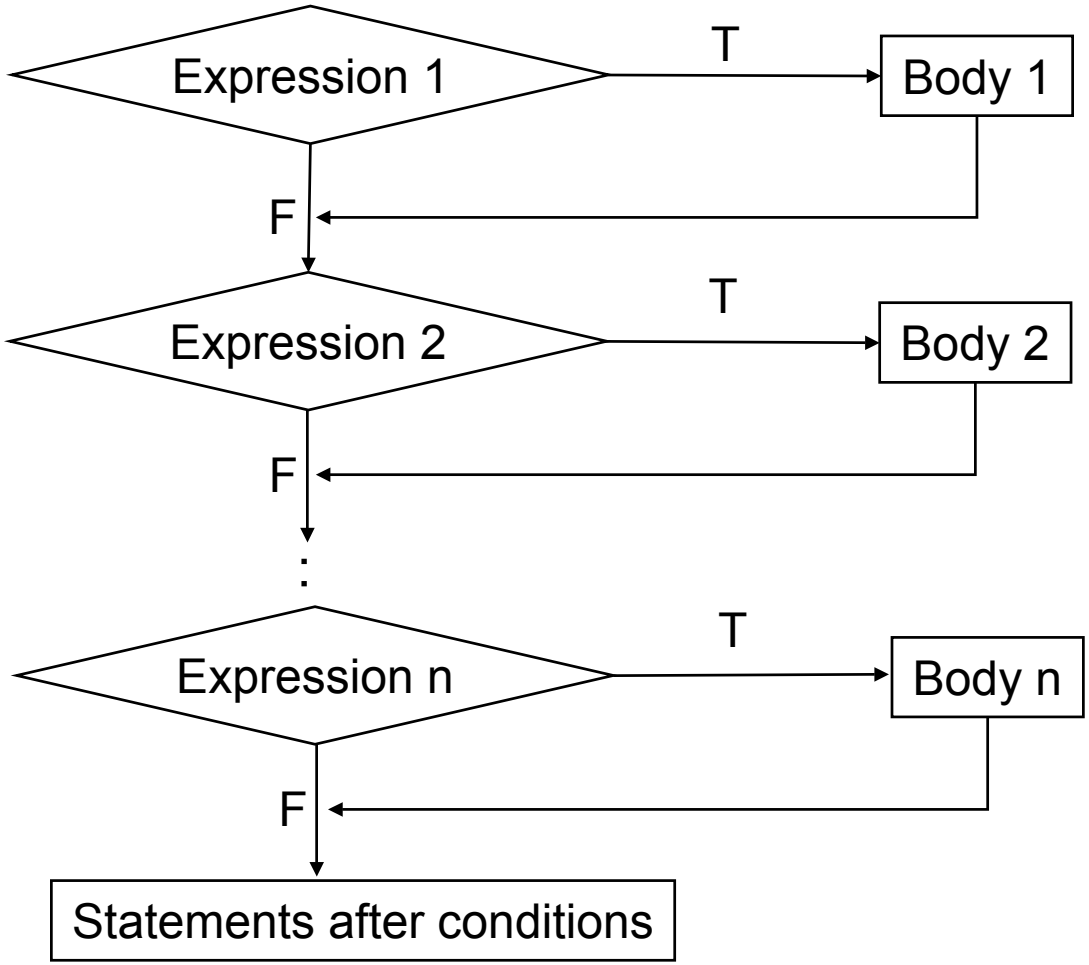
Multiple If's: Non-Exclusive Conditions

Any, all or none of the conditions may be true

Format:

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

Multiple If's:Flowchart



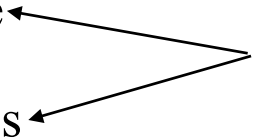
Multiple If's: Non-Exclusive Conditions (Example)

```
if (x > 0) then

    writeln('X is positive);

if (y > 0) then

    writeln('Y is positive');

If (z > 0) then

    writeln('Z is postive');
```

Multiple If's: Exclusive Conditions

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)

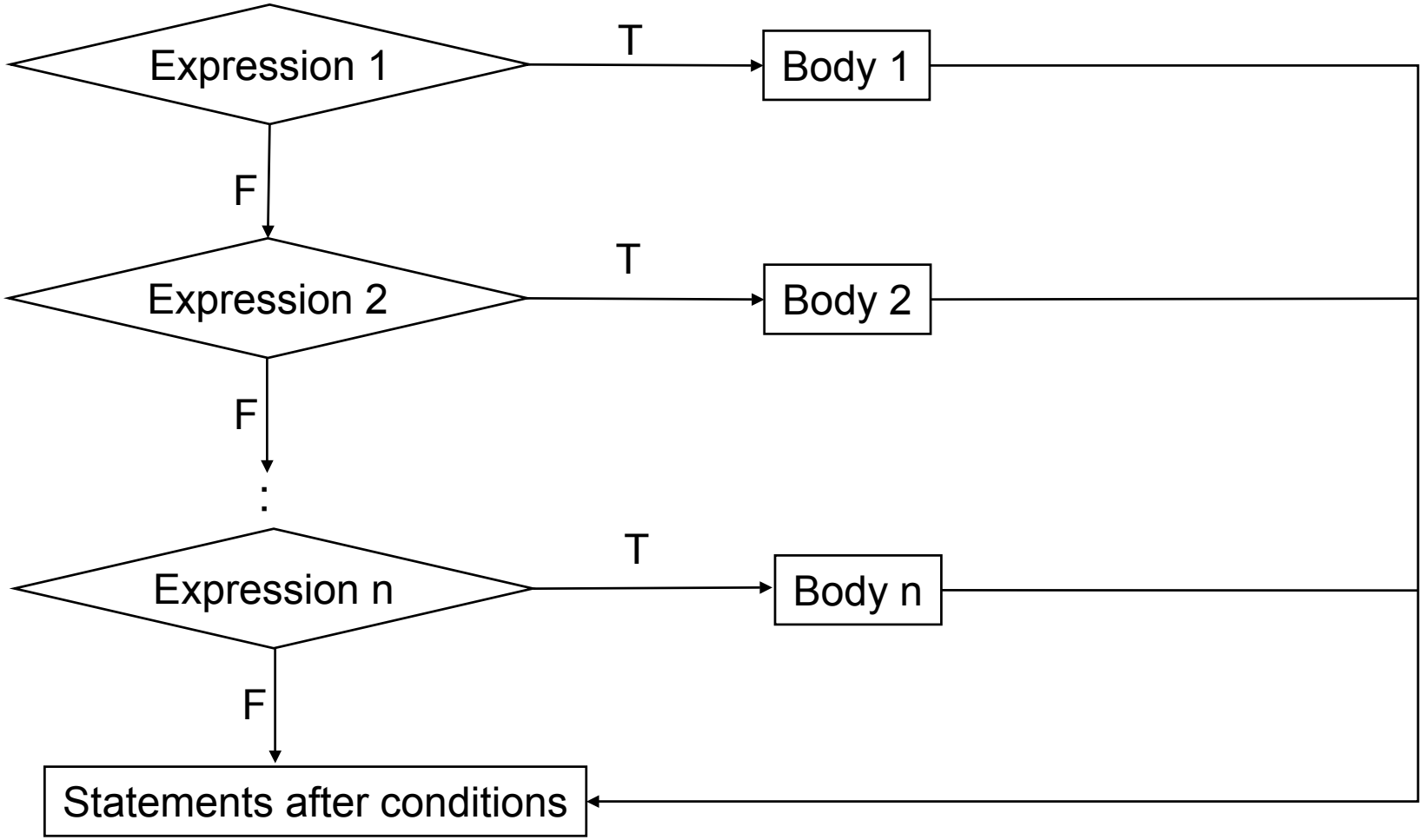
```
if (gpa = 4) then
    letter := 'A';
if (gpa = 3) then
    letter := 'B';
if (gpa = 2) then
    letter := 'C';
if (gpa = 1) then
    letter := 'D';
if (gpa = 0) then
    letter := 'F';
```

Multiple If, Else-If's: Mutally 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;
```

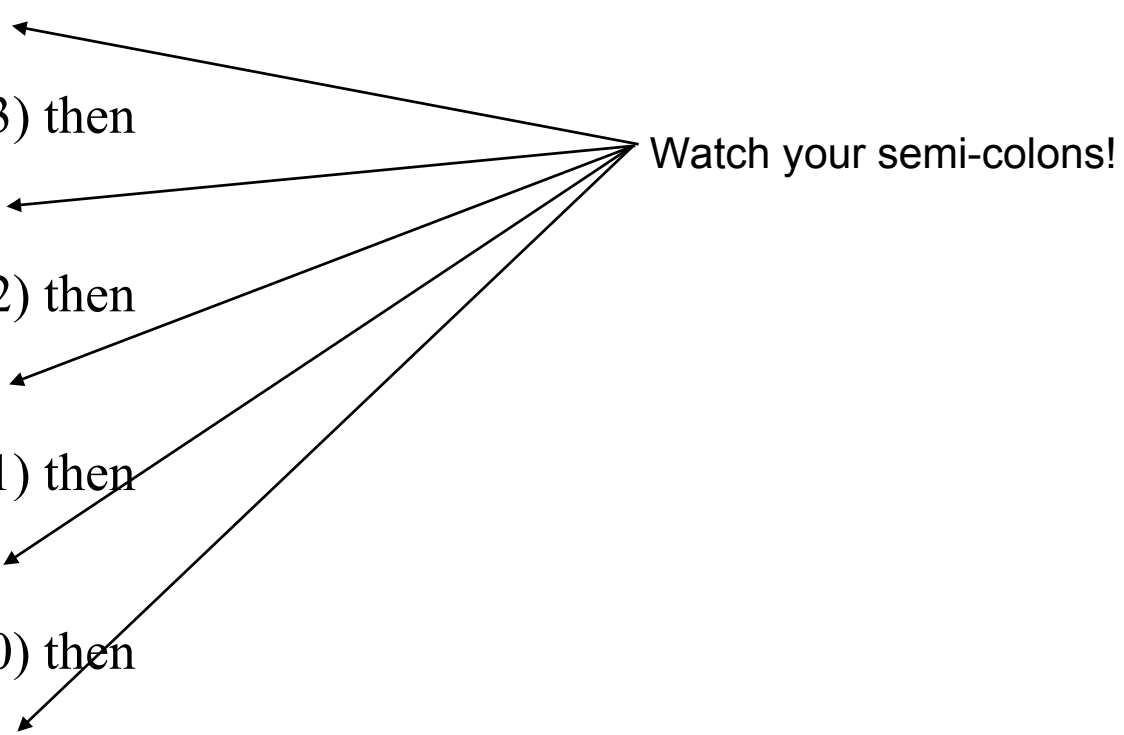
Multiple If, Else-If's: Flowchart



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

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

```
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';
```



Case Statements

An alternative to the if, else-if (only one condition is true)

Format (integer):

```
Case (expression) of
    i1:
        body;
    i2:
        body;
     :
    in:
        body;
    otherwise:
        body;
```

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:
        begin
            writeln('You got an A');
        end; (* GPA of 4 *)
    3:
        begin
            writeln('You got a 'B');
        end; (* GPA of 3 *)
```

Case Statements: Integer Example (2)

```
2:
    begin
       writeln('You got a C');
    end; (* GPA of 2 *)
1:
    begin
        writeln('You got a D');
    end; (* GPA of 1 *)
0:
    begin
        writeln('You got an F');
    end; (* GPA of 0 *)
end; (* case *)
```

Case Statements: Characters

Format (char):

Case (expression) of

‘c_1’:

body;

‘c_2’:

body;

:

‘c_n’

body;

otherwise:

body;

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':
        begin
            writeln('GPA = 4');
        end; (* GPA of 4 *)
    'B':
        begin
            writeln('GPA = 3');
        end; (* GPA of 3 *)
```

Case Statements: Character Example (2)

```
'C':
    begin
        writeln('GPA = 2');
    end; (* GPA of 2 *)
'D':
    begin
        writeln('GPA = 1');
    end; (* GPA of 1 *)
'F':
    begin
        writeln('GPA = 0');
    end; (* GPA of 0 *)
end; (* case *)
```

Summary

How is decision making implemented via Pascal constructs:

- If-then
- If-then-else
- Case-of

What are Boolean expressions and what are valid operators and operands?

How to handle simple vs. multiple statements in the body of a decision-making statement.

What are compound Boolean expressions?

How does nested decision making work?

Exclusive vs. non-exclusive alternatives when making decisions.