Chapter 4

Variables
The three attributes of a variable

- A name
- A type
- A value
MODULE Pbox04A;
IMPORT StdLog;

PROCEDURE Rectangle*;
VAR
  width: REAL;
  length: REAL;
BEGIN
  width := 3.6;
  length := 12.4;
  StdLog.String("The width is "); StdLog.Real(width); StdLog.Ln;
  StdLog.String("The length is "); StdLog.Real(length); StdLog.Ln
END Rectangle;

END Pbox04A.

The width is 3.6
The length is 12.4
<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
</tbody>
</table>

**Figure 4.2**
The real operators.
MODULE Pbox04B;
  IMPORT StdLog;

  PROCEDURE Rectangle*;
    VAR
      width, length: REAL;
      area, perim: REAL;
    BEGIN
      width := 3.6;
      length := 12.4;
      StdLog.String("The width is "); StdLog.Real(width); StdLog.Ln;
      StdLog.String("The length is "); StdLog.Real(length); StdLog.Ln;
      area := width * length;
      perim := 2.0 * (width + length);
      StdLog.String("The area is "); StdLog.Real(area); StdLog.Ln;
      StdLog.String("The perimeter is "); StdLog.Real(perim); StdLog.Ln
    END Rectangle;

END Pbox04B.

The width is 3.6
The length is 12.4
The area is 44.64
The perimeter is 32.0
MODULE Pbox04C;
   IMPORT StdLog;

   PROCEDURE Change*;
      VAR
         cents: INTEGER;
      BEGIN
         cents := 39;
         StdLog.String("You have "); StdLog.Int(cents);
         StdLog.String(" cents in change."); StdLog.Ln
      END Change;
   END Pbox04C.

   You have 39 cents in change.

Figure 4.4
A procedure that sets the value of an integer variable and outputs it to the Log.
MODULE Pbox04D;
  IMPORT StdLog;

  PROCEDURE Error;
    VAR
      i: INTEGER;
    BEGIN
      i := 2.7;
      StdLog.String("The value of i is "); StdLog.Int(i); StdLog.Ln
    END Error;

END Pbox04D.

Figure 4.5
A procedure that tries to assign a real value to an integer variable. This procedure has a bug.
### Figure 4.6
The integer operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>DIV</td>
<td>Division</td>
</tr>
<tr>
<td>MOD</td>
<td>Modulo</td>
</tr>
</tbody>
</table>
- $m \text{ div } n$ is the quotient of $m \div n$.
- $m \text{ mod } n$ is the remainder of $m \div n$.

Let $q$ represent the quotient and $r$ represent the remainder, so that

\[
q = m \text{ div } n \\
r = m \text{ mod } n
\]

Then the relationship between div and mod is expressed mathematically as

\[
m = q \cdot n + r \quad 0 \leq r < n
\]
MODULE Pbox04E;
 IMPORT StdLog;

PROCEDURE MakeChange*;
 VAR
   cents: INTEGER;
   dimes, nickels, pennies: INTEGER;
BEGIN
   cents := 39;
   StdLog.String("You have "); StdLog.Int(cents);
   StdLog.String(" cents in change."); StdLog.Ln;
   dimes := cents DIV 10;
   cents := cents MOD 10;
   nickels := cents DIV 5;
   pennies := cents MOD 5;
   StdLog.String("Dimes: "); StdLog.Int(dimes); StdLog.Ln;
   StdLog.String("Nickels: "); StdLog.Int(nickels); StdLog.Ln;
   StdLog.String("Pennies: "); StdLog.Int(pennies); StdLog.Ln
END MakeChange;

END Pbox04E.

You have 39 cents in change.
Dimes: 3
Nickels: 1
Pennies: 4

Figure 4.7
The number of dimes, nickels, and pennies required for a given amount of change.
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC(v)</td>
<td>v := v + 1</td>
</tr>
<tr>
<td>INC(v, n)</td>
<td>v := v + n</td>
</tr>
<tr>
<td>DEC(v)</td>
<td>v := v - 1</td>
</tr>
<tr>
<td>DEC(v, n)</td>
<td>v := v - n</td>
</tr>
</tbody>
</table>

**Figure 4.8**
The increment and decrement functions for integers.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Type of operands</th>
<th>Type of result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>Both integer</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one real</td>
<td>Real</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>Both integer</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one real</td>
<td>Real</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>Both integer</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one real</td>
<td>Real</td>
</tr>
<tr>
<td>/</td>
<td>Real division</td>
<td>Integers or reals</td>
<td>Real</td>
</tr>
<tr>
<td>DIV</td>
<td>Integer division</td>
<td>Integers</td>
<td>Integer</td>
</tr>
<tr>
<td>MOD</td>
<td>Modulus</td>
<td>Integers</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Figure 4.9
Types of results for the arithmetic operations.
DEFINITION Math;

PROCEDURE Pi (): REAL;
PROCEDURE Sqrt (x: REAL): REAL;
PROCEDURE Exp (x: REAL): REAL;
PROCEDURE Ln (x: REAL): REAL;
PROCEDURE Log (x: REAL): REAL;
PROCEDURE Power (x, y: REAL): REAL;
PROCEDURE IntPower (x: REAL; n: INTEGER): REAL;

PROCEDURE Sin (x: REAL): REAL;
PROCEDURE Cos (x: REAL): REAL;
PROCEDURE Tan (x: REAL): REAL;
PROCEDURE ArcSin (x: REAL): REAL;
PROCEDURE ArcCos (x: REAL): REAL;
PROCEDURE ArcTan (x: REAL): REAL;

END Math.

Figure 4.10
Some of the math functions from the interface of the Math module.
Figure 4.11
The number line for some of the character values.
DEFINITION PboxStrings;

  PROCEDURE Lower (ch: CHAR): CHAR;
  PROCEDURE Upper (ch: CHAR): CHAR;
  PROCEDURE ToLower (from: ARRAY OF CHAR; OUT to: ARRAY OF CHAR);
  PROCEDURE ToUpper (from: ARRAY OF CHAR; OUT to: ARRAY OF CHAR);
  PROCEDURE IntToString (n, minWidth: INTEGER; OUT s: ARRAY OF CHAR);
  PROCEDURE RealToString (x: REAL; minWidth, dec: INTEGER; OUT s: ARRAY OF CHAR);

END PboxStrings.
MODULE Pbox04F;
  IMPORT StdLog;

  PROCEDURE PrintString*;
    VAR
      message: ARRAY 128 OF CHAR;
    BEGIN
      message := "What's up, Doc?";
      StdLog.String(message); StdLog.Ln
    END PrintString;
  END PrintString;

END Pbox04F.

Figure 4.13
A procedure that declares a variable with string type.

Figure 4.14
Storage of a string value in an array of characters.
MODULE Pbox04G;
    IMPORT StdLog, PboxStrings;

    PROCEDURE Change*;
        VAR
            cents: INTEGER;
            centString: ARRAY 16 OF CHAR;
            message: ARRAY 64 OF CHAR;
        BEGIN
            cents := 39;
            PboxStrings.IntToString(cents, 1, centString);
            message := "You have " + centString + " cents in change."
            StdLog.String(message); StdLog.Ln
        END Change;

    END Pbox04G.

Figure 4.15
A procedure that uses the + operator to concatenate strings. It imports the PboxStrings module.

Figure 4.16
The value of centString.

3 9 0X
Figure 4.17
Character array assignments.
<table>
<thead>
<tr>
<th>CP</th>
<th>GCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>( \mathbb{Z} )</td>
</tr>
<tr>
<td>REAL</td>
<td>( \mathbb{R} )</td>
</tr>
</tbody>
</table>

**Figure 4.18**
Specifying type in GCL.