Subprograms

Procedures

Recall - Functions

A function is a subprogram that accepts input parameters and then returns an output value. The input values and output value can be the same or different data types.

For example:

- a pop machine inputs money and selections on a keypad, and outputs a type of soft drink
- the `length()` function has a string for input, and the output is an integer.

Functions

Important Properties:
1. Input parameters can be of different data types.

2. There must be a single output value (result), and it can also be of a different data type than the inputs.

3. The input parameters are not changed by the function.

Functions vs. Procedures

When first learning about functions and procedures, they are often interchangeable. Both can accomplish simple tasks equally well.

1. A procedure does not have a return value. A function must return a single value.

2. A procedure can make changes to input parameters, but only if the procedure is declared to allow it. This allows a procedure to return multiple values by making changes.

Declaring Procedures

```
function doThis ( input1 : data1 ) : data2
   ...
end doThis

procedure doThat ( input1 : data1 )
   ...
end doThat
```

Notice that the only difference (so far), is that the ": data2" is missing from the end.

Example – Perimeter & Area of a Circle

We have already considered this problem using functions. Like a function, a procedure must be declared before we can use it.

It is also important to note that declaring a procedure does not do anything on its own. It must be called from the main program for the code to be run.

Declaring Procedures - Example

```
function circumference(radius : real) : real
   % returns circumference of a circle
   result 2 * 3.14 * radius
end circumference

function circleArea(radius : real) : real
   % returns area of a circle = pi * r-squared
   result 3.14 * radius * radius
end circleArea

procedure circleStats(radius : real)
    % outputs perimeter and area of circle
    put "P = ", circumference(radius)
    put "A = ", circleArea(radius)
end circleStats
```

Calling Procedures

Since a procedure does not have a return value, there is no need for the assignment operator ":="

To use a function, we might code

```
newValue := someFunction (input1, input2)
```

For a procedure, no assignment is necessary

```
someProcedure (input3, input4)
```

Calling a Procedures - Example

```
procedure circleStats(radius : real)
    % outputs perimeter and area of circle
    put "P = ", circumference(radius)
    put "A = ", circleArea(radius)
end circleStats

%%%%% Main Program %%%%%
var radius : real
put "Enter a radius"..
get radius

circleStats (radius)
```