§5.4-5.8: FOR Loops, More Arrays

•devo
•midterms back

Reminders:

1) journals in folder

12 Oct 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Review of types

- Modula-2 allows us to define our own types in addition to the built-in types we've been using so far:
  - Atomic types
    - Scalar types
      - **Real types (REAL, LONGREAL)**
      - **Ordinal types**
        - **Whole number types (INTEGER, CARDINAL)**
        - **Enumerations (5.2.1)**
        - **Subranges (5.2.2)**
  - Structured (aggregate) types
    - Arrays (5.3)
      - **Strings (5.3.1)**
    - Sets (9.2-9.6)
    - Records (9.7-9.12)

done

done

What's on for today (5.4-5.8)

- FOR loops
  - Loop control variable
    - Needs initialization?
    - Value after the loop?
  - FOR vs. WHILE: pros/cons?
- Arrays as procedure parameters
  - Type compatibility for value/variable params
  - Open arrays
    - HIGH
- Multidimensional arrays

Iterating

- We have often used counters to iterate in a loop

```
counter := start;
WHILE counter <= stop
    DO
        statement sequence;
        INC (counter);
    END;
```

- Modula-2 provides a shorthand to help:

```
FOR counter := start TO stop
    DO
        statement sequence;
    END;
```

Loop control variable

- The loop control variable (e.g., counter)
  - Can be any ordinal type (enumerations, etc.)
  - Does not need to be initialized before the loop
  - Value is undefined after the loop:

```
FOR counter := start TO stop
    DO
        statement sequence;
    END;
counter := 0;
```

    - (can't depend on counter having a particular value)

Increments

- An optional constant increment can be given:

```
FOR counter := start TO stop BY increment
    DO
        statement sequence;
    END;
```

  - This is equivalent to using

    ```
    INC (counter, increment)
    ```

  - The increment can be negative, too
  - The increment must be a constant expression
  - Must be whole number type (not enumeration):

    ```
    FOR today := Mon TO Fri BY 1
    ```

FOR as shorthand for WHILE

- For most iterative loops, FOR is a good shorthand
- But WHILE gives you more control:
  - e.g., exiting a loop early:

```
counter := 0;
WHILE counter < max
   DO
      IF (user wants to quit early)
         THEN
            counter := max;
         END;
      INC (counter);
   END;
```

- Loop control variable may not be threatened inside FOR

FOR loops and arrays

- Find average of an array:

```
CONST
    length = 10;
VAR
    myArray : ARRAY [1 .. length] OF REAL;
    sum, average : REAL;
    index : CARDINAL;
BEGIN
    sum := 0;
    FOR index := 1 TO length
        DO
            INC (sum, myArray [index]);
        END;
    average := sum / length;
```

Arrays as parameters

```
PROCEDURE Average
    (myArray : ARRAY [1 .. 10] OF REAL) : REAL;
VAR
    sum : REAL;
    index : CARDINAL;
BEGIN
    sum := 0;
    FOR index := 1 TO length
        DO
            INC (sum, myArray [index]);
        END;
    RETURN sum / length;
END Average;
```

- But this function can only take arrays of size 10!

Array type compatibility

- When value parameters use array types:
  - Actual param and formal param must be assignment compatible
- When variable parameters use array types:
  - Actual param and formal param must be exactly the same

Open arrays

- An open array does not specify the range:

```
PROCEDURE Average
    (myArray : ARRAY OF REAL) : REAL;
```

- A REAL array of any length is compatible
- Find the length of the array with HIGH:

```
FOR index := 0 TO HIGH (myArray)
```

- Indexing of open arrays is always

```
[ 0 .. HIGH (myArray) ]
```

  - Even if the array is usually indexed by enumeration

Multidimensional arrays

- Multidimensional arrays are simply arrays of arrays:

  ```
  MatrixA : ARRAY [1 .. 3] OF ARRAY [1 .. 4] OF REAL;
  ```

  - Shorthand:

  ```
  MatrixA : ARRAY [1 .. 3], [1 .. 4] OF REAL;
  ```

- Accessing:

  ```
  MatrixA [2] [3] := 2.3;
  ```

  - Shorthand:

  ```
  MatrixA [2, 3] := 2.3;
  ```

- Row-major convention:

$$\begin{array}{cl} \begin{pmatrix} 1.1 & 1.2 & 1.3 & 1.4 \\ 2.1 & 2.2 & \boxed{2.3} & 2.4 \\ 3.1 & 3.2 & 3.3 & 3.4 \end{pmatrix} & \begin{array}{l} \\ \leftarrow \text{MatrixA[2]} \\ \\ \end{array} \end{array}$$

Open multidimensional arrays

- Multidimensional arrays can also be open:

```
PROCEDURE Average
    (A : ARRAY OF ARRAY OF REAL) : REAL;
VAR sum : REAL; row, col : CARDINAL;
BEGIN
    sum := 0;
    FOR row := 0 TO HIGH (A)
        DO
            FOR col := 0 TO HIGH (A [row])
                DO
                    INC (sum, A [row, col]);
                END;
        END;
    RETURN sum / ( HIGH (A) * HIGH (A[0]) );
END Average;
```

Number of rows — HIGH (A)

Number of columns — HIGH (A [row]), HIGH (A[0])

Review of today (5.4-5.8)

- FOR loops
  - Loop control variable
    - Needs initialization?
    - Value after the loop?
  - FOR vs. WHILE: pros/cons?
- Arrays as procedure parameters
  - Type compatibility for value/variable params
  - Open arrays
    - HIGH
- Multidimensional arrays

TODO items

- Lab5 due next week:
  - §6.11 #(25 / 33) (choose one)
- Homework: §5.11 #22 due Friday
- Quiz ch5 this Friday!
- Reading: through §6.3 for Friday