MPL Iteration Structures

MPL contains two iteration structures that allow for repeated evaluation of a sequence of statements. The first iteration structure is the **while** structure which has the general form

$$\begin{array}{l} \textbf{while} \;\; \textit{condition} \; \textbf{do} \\ \quad S_1; \\ \quad S_2; \\ \quad \;\vdots \\ \quad S_n; \end{array} \tag{1}$$

where *condition* is a logical (or relational) expression. This structure is evaluated by first evaluating *condition*, and if it is to **true**, the indented statements $S_1, S_2, \ldots, S_m$ are evaluated. Once this is done, the process repeats, and again if the logical *condition* is **true**, the indented statements are evaluated. The process continues in this way checking if *condition* is **true** and if so, evaluating the indented statements. On the other hand once *condition* evaluates to **false**, the indented statements are not evaluated, and the structure terminates.

Example 1 The sum of the first $n+1$ terms of a Taylor series for a function $u(x)$ about $x = a$ is given by

$$\sum_{i=0}^{n} \frac{u^{(i)}(a)}{i!}(x-a)^i \tag{2}$$

where $u^{(i)}$ is the ith derivative of $u(x)$, and $u^{(0)} = u(x)$. When n is a non-negative integer, (2) is obtained with the following **MPL** statements.

```
1   i := 1;
2   s := Substitute(u, x = a);
3   while  i ≤ n  do
4       u := Derivative(u, x);
5       s := s + Substitute(u, x = a)/i! * (x − a)^i;
6       i := i + 1;
```

The substitution in line 2 initializes s to $u^{(0)}(a) = u(a)$, and each traversal through the **while** loop adds one additional term of the Taylor series to s

and increases the counter i by 1. Eventually $i = n+1$, and so the condition $i \le n$ is **false**, and the **while** structure terminates.

For example, if $u = \sin(x)$, $n = 3$, and $a = 0$, after executing the loop we obtain $s = x - x^3/6$.

The second iteration structure is the **for** structure which has the general form

$$\begin{array}{l} \textbf{for}\ \ i := \textit{start}\ \textbf{to}\ \textit{finish}\ \textbf{do} \\ \quad S_1; \\ \quad S_2; \\ \quad \ \vdots \\ \quad S_n; \end{array} \tag{3}$$

where i is a variable and *start* and *finish* are expressions that evaluate to integer values. When $\textit{start} \le \textit{finish}$, the indented statements are evaluated for each integer value of $i = \textit{start}, \textit{start}+1, \ldots, finish$. If $\textit{start} > finish$, the indented statements are not evaluated[1].

Example 2 The sum of the first $n+1$ terms of the Taylor series can also be obtained using a **for** structure:

$$\begin{array}{l} s := \textit{Substitute}(u, x = a); \\ \textbf{for}\ \ i := 1\ \textbf{to}\ n\ \textbf{do} \\ \quad u := \textit{Derivative}(u, x); \\ \quad s := s + \textit{Substitute}(u, x = a)/i! * (x - a)^i; \end{array}$$

All computer algebra languages provide iteration structures.

MPL	Maple	Mathematica	MuPAD
for	`for`	`For`	`for`
while	`while`	`While`	`while`

Figure 1. MPL iteration structures and the corresponding operators in Maple, Mathematica, and MuPAD. (Implementation: Maple (mws), Mathematica (nb), MuPAD (mnb).)

Return to Chapter 1, page 3

[1] Some of our procedures contain **For** loops that include a *Return* statement. In this case, we intend that both the loop and the current procedure terminate when the *Return* is encountered, and that the value returned by the procedure is the value of the operand of the *Return* statement. The `for` statements in both Maple and MuPAD work in this way. However, in Mathematica, a `Return` in a `For` statement will only work in this way if the upper limit contains a relational operator (e.g., `i<=N`). (Implementation: Mathematica (nb).)