Приложение C. Цикл Дейкстры

Дейкстра [C.1] (см. также [C.2]) ввел многоветочный вариант цикла WHILE, обосновав его в своей теории систематического вывода императивных программ. Этот цикл оказывается удобным для выражения и, главное, верификации алгоритмов, обычно записываемых через вложенные циклы, позволяя резко снизить усилия по отладке.

В версии языка Оберон, известной как Оберон-07 и представленной в [C.3], синтаксис этого цикла определяется следующим правилом:

```
WhileStatement
=  WHILE логическое выражение DO
     последовательность операторов
   {ELSIF логическое выражение DO
     последовательность операторов}
   END.
```

Если вычисление любого из логических выражений (охран) дает TRUE, то выполняется соответствующая последовательность операторов. Вычисление охран и выполнение соответствующих последовательностей повторяется до тех пор, пока все охраны не будут давать FALSE. Таким образом, постусловием для этого цикла является конъюнкция отрицаний всех охран.

Пример:

```
WHILE m > n DO m := m - n
ELSIF n > m DO n := n - m
END
```

Постусловие: ~(m > n) & ~(n > m), что эквивалентно n = m.

Инвариант цикла должен выполняться в начале и в конце каждой ветви.

Грубо говоря, обычно n-веточный цикл Дейкстры соответствует конструкциям из n обычных циклов, каким-то образом вложенных друг в друга.

Удобство цикла Дейкстры обусловлено тем, что вся логика сколь угодно сложного цикла выводится на один уровень и радикально проясняется, причём алгоритмы с разными случаями запутанных циклов трактуются совершенно единообразно.

Эффективный способ построения такого цикла состоит в том, чтобы перечислять все возможные ситуации, которые могут возникнуть, описывая их соответствующими охранами, и для каждой из них независимо от остальных строить операции, обеспечивающие продвижение по данным, вместе с операциями, обеспечивающими восстановление инварианта. Перечисление ситуаций заканчивается, когда дизъюнкция (или) всех охран покроет предполагаемое предусловие цикла. При этом задача корректного построения цикла облегчается, если отложить беспокойство о порядке вычисления охран и выполнения ветвей, экономии операций при вычислении охран и т.п. до тех пор, пока цикл не будет корректно построен. Такие мелкие оптимизации редко по-настоящему важны, а их реализация сильно облегчается, когда корректность сложного цикла уже обеспечена.

Хотя в теории Дейкстры последовательность выбора ветвей цикла и вычисления соответствующих охран не определена, в этой книжке принято, что охраны вычисляются в текстуальном порядке.

Во многих языках программирования цикл Дейкстры приходится моделировать. В старых версиях Оберона, включая Компонентный Паскаль, достаточно использовать цикл LOOP, тело которого состоит из многоветочного условного оператора с оператором выхода в ветке ELSE:

LOOP IF *логическое выражение* THEN
  последовательность операторов
{ELSIF *логическое выражение* THEN
  последовательность операторов}
ELSE EXIT END END.

В других языках конструкция может быть более громоздкой, но это с лихвой окупается упрощением построения и отладки сложных циклов.

Литература

[C.1] E.W. Dijkstra. A Discipline of Programming. Prentice-Hall, 1976. (Имеется перевод: Дейкстра Э. Дисциплина программирования. М: Мир, 1978.)

[C.2] D. Gries. The Science of Programming. Springer-Verlag, 1981. (Имеется перевод: Д. Грис. Наука программирования. М: Мир, 1984.).

[C.3] N. Wirth. The Programming Language Oberon. Revision 1.9.2007.