А. А. Вылиток

Выражения с побочными эффектами. Язык Паскаль

Программа, процедура, функция в языке Паскаль состоят из заголовка (содержащего соответственно ключевые слова ***program***, ***procedure***, ***function***) и *блока*. Блок содержит раздел описаний (меток, констант, типов, переменных, процедур и функций) и раздел операторов.

Значения переменных изменяются операторами ввода, присваивания, вызовами процедур. Операторы могут содержать выражения [1]. Основное назначение выражения — получить новое значение из уже известных значений путем выполнения определенных операций.

Образно говоря, выражение — это фабрика по производству значений. Сырьем для этой фабрики являются значения переменных, входящих в выражение, а продукцией — получаемое значение. Продукция потребляется оператором, внутри которого данное выражение находится. Рассмотрим «работу» выражения *a* + 3. Пусть *a* имеет значение 2. Прежде всего, вычисляются значения простейших подвыражений *a* и 3 — они равны соответственно двум и трем, — а затем конструируется новое значение, равное их сумме. Таким образом, фабрика изготовит нам целое число пять.

Заметим, что не все тонкости производственных процессов фабрики нам известны. К примеру, про выражение 1*3 + 4 ***div*** 2 мы можем сказать, что операция «+» будет выполнена позже, чем любая из операций «*» и «***div***», поскольку операндами операции «+» являются подвыражения 1*3 и 4 ***div*** 2. Но мы не можем ничего сказать о порядке «*» и «***div***», поскольку в Паскале порядок вычисления операндов для операции «+» не определен [2]. Это значит, что имеется некоторая свобода: выполнить сначала операцию «*», затем «***div***», или сначала «***div***», затем «*» или, если позволяет вычислительная система, выполнить «*» и «***div***» одновременно (параллельно). Любой из этих трех способов может быть выбран для вычисления — значение выражения 1*3 + 4 ***div*** 2 от этого выбора не зависит. Фабрика не должна оказывать влияние на окружающую среду, т. е. не должна «портить» (изменять) значения переменных. Иначе свободный выбор последовательности операций может повлиять на готовую продукцию и получится не то значение, которое ожидалось. Ниже будет приведен пример, подтверждающий такую возможность.

Выражение обладает *побочным эффектом*, если в процессе его вычисления изменяются значения переменных, описанных в том же блоке, где находится данное выражение, или в объемлющих блоках. Выполнение стандартных операций (сложение, умножение и т.п.) не приводит к побочным эффектам. Однако выражение может содержать вызов функции [3]. Если вызванная функция изменяет только свои локальные переменные, то такой вызов не обладает побочным эффектом (локальные переменные

---

[1] Выражение в языке Паскаль не является самостоятельным действием, оно всегда находится внутри какого-нибудь оператора, например, в правой части оператора присваивания, в заголовке цикла ***while*** и др.

[2] Порядок вычисления операндов не определен для любой бинарной операции. Также не определен порядок вычисления фактических параметров в вызове функции.

[3] Вызов любой стандартной функции (*sin*, *cos*, *abs* и т. п.) не обладает побочным эффектом при условии, что фактический параметр задан выражением без побочных эффектов.

могут меняться, но они существуют, только пока функция выполняется, и «исчезают», когда функция возвращает значение в точку вызова). Если же функция изменяет глобальную (внешнюю) по отношению к ней переменную, то вызов такой функции обладает побочным эффектом.[4]

Побочные эффекты в выражениях Паскаля считаются плохим стилем, но иногда на это идут сознательно, желая повысить эффективность программы. Например, можно описать логическую функцию *find* (*A, e, num*), определяющую содержится ли в массиве *A* элемент, равный *e*, так, чтобы в случае успеха индекс найденного элемента присваивался бы переменной *num*.

Рассмотрим следующую программу.

```
program bad_programming(output);
var i: integer;
    a: array[1..2] of integer;
function foo: integer;
begin
  foo := i;
  i   := i+1
end;
begin
  i:=1; a[1]:=0; a[2]:=0;
  a[foo]:=foo;
  writeln(a[1],a[2]);
  writeln(foo-foo-foo)
end.
```

Так как порядок вычисления левой и правой частей оператора присваивания и порядок вычисления левого и правого операнда бинарной операции (здесь «−» — вычитание) зависят от реализации, нельзя сказать, что будет напечатано данной программой. С точки зрения Стандарта такая программа должна считаться ошибочной. Побочным эффектом в данной программе обладает выражение *foo*, а также выражения, содержащие его в качестве подвыражения.

Иногда словосочетание «побочный эффект» трактуют более широко. Например, говорят о побочном эффекте процедуры сортировки массива *sort* (*A*), которая не только переупорядочивает элементы массива *A* по неубыванию, но и увеличивает значение глобальной переменной *i* на единицу, о чем никак нельзя догадаться по заголовку процедуры. Побочного эффекта в строгом смысле здесь нет, поскольку вызов процедуры — это не операция (не выражение), а оператор. Одно из предназначений операторов как раз и состоит в том, чтобы изменять значения переменных. Другой вопрос, легко ли читателю программы понять, какие переменные будут меняться при выполнении оператора вызова процедуры. Наверное, об этом должен позаботиться автор программы...

[4] Если функция осуществляет ввод или вывод, то вызов такой функции обладает побочным эффектом, поскольку изменяются глобальные файловые переменные *input* или *output*.