Б. Я. Львин

МАКРОСРЕДСТВА, АДЕКВАТНЫЕ ЯЗЫКУ МОДУЛА-2

В системе программирования Модула-2 часть вычислений выполняется на этапе трансляции исходной программы, но она ограничена вычислением константных выражений. При недостатке памяти и при возможности использовать ПЗУ, где удобно хранить константы, желательно максимально возможную часть вычислений выполнить на этапе трансляции и получить эффективную остаточную программу [1]. Представляется целесообразным разрешение использовать не только выражения, но и операторы периода генерации. Это дает возможность употреблять в программах именованные константы сложных типов и легко производить настройку программы на конкретные условия применения.

По предложению Мак-Илроя [2], важно, чтобы средства периода генерации совпадали с обычными средствами языка (так, в языке Модула-2, за исключением вызова функций, совпадает запись общих и константных выражений). В то же время из соображений эффективности трансляции действия

периода компиляции не должны отделяться от остаточной программы по усмотрению транслятора, как это принято при смешанных вычислениях [1], а должны явно указываться в тексте.

В качестве одного из решений можно ввести в язык макромодули, синтаксис которых совпадает с обычными модулями, но все предписанные действия выполняются при трансляции программы. В этом случае в обычных модулях переменные макромодулей (макропеременные) должны выступать в роли констант, а не переменных с начальными значениями. Последние не эффективны, особенно в случае ПЗУ. В обычных модулях доступны также экспортируемые макромодулями константы и типы. Использование же процедур из макромодулей (макроопределений) и присваивание значений макропеременным возможно только в макромодулях. В макромодулях естественно запретить использование каких-либо объектов обычных модулей. Тогда объявление констант в обычных модулях можно рассматривать как сокращенное обозначение для макромодуля, в котором вычисляется значение экспортируемой макропеременной.

Это решение существенно усложняет транслятор, поскольку текст программного модуля должен обрабатываться в два этапа: сначала интерпретация всех макромодулей (возможно, вложенных), а затем – трансляция обычных. Для упрощения этих проблем естественно потребовать, чтобы макромодули не были вложены в обычные модули и процедуры и наоборот. Тем самым исполнение в период генерации или трансляция с последующим исполнением становится особенностью единицы компиляции.

В языке Модула-2 есть единицы компиляции, которые используются только на этапе трансляции – модули определений. В них задаются те константы, типы (возможно, скрытые), переменные и заголовки процедур, которые доступны извне соответствующего модуля реализации. Для выполнения действий по макрогенерации в модулях определений необходимо, кроме того, допустить:

– объявления макропеременных, связывающих период генерации, когда они переменные, и период исполнения, когда они константы указанного типа (нескрытого);

– макроопределения, задающие процедуры периода генерации, которые могут быть использованы как в этом модуле определений, так и экспортированы в другие;

– тело с операторами, задающими выполняемые модулем действия периода генерации.

Предлагаемый синтаксис приведен на рис. 1.

```
МодульОпределений = DEFINITION MODULE идент ";"
  {импорт} {определение} [BEGIN ПослОператоров] END ".".
определение = CONST {ОбъявлениеКонстанты ";"} |
  TYPE {идент ["=" тип] ";"} |
  VAR {ОбъявлениеПеременной ";"} |
  ЗаголовокПроцедуры ";" |
  MACRO VAR {ОбъявлениеПеременной ";"} |
  MACRO ОбъявлениеПроцедуры ";" .
```

Рис. 1

При таких макросредствах интерпретация операторов языка Модула-2 требуется только при трансляции модулей определений, когда не нужна генерация объектного кода. В создаваемый при этом символьный файл [3] необходимо дополнительно помещать информацию о макроопределениях, тем самым создавая макробиблиотеку и результирующие значения макропеременных. В этом случае окончательные значения макропеременных фиксируются при трансляции включающего их модуля определений; для остальных модулей (даже определений) они константы. Это существенно при написании макроопределений, которые при использовании в содержащем их модуле могут устанавливать значения его макропеременных, а в импортирующих модулях – только использовать эти значения.

Имеет смысл распределить память на период исполнения под константы сложных типов, получившиеся из макропеременных, подобно памяти под глобальные переменные модуля только в заполненной секции (такой, как в настоящее время имеется для строк и дескрипторов массивов). Только сделать это надо при трансляции модуля определений и передать эту информацию через символьный файл. Это позволит иметь в скомпонованной программе один экземпляр сложной константы для всех использующих ее единиц компиляции (а не в каждой, как в настоящее время для строк) (рисунки 2, 3).

```
DEFINITION MODULE POLINOM;
   MACRO VAR P: ARRAY [0..5] OF REAL;
   PROCEDURE POL(X: REAL): REAL;
BEGIN P[0]:= 1.2; P[1]:= 2.3; P[2]:= 3.4;
      P[3]:= 4.5; P[4]:= 5.6; P[5]:= 6.7;
END POLINOM.
IMPLEMENTATION MODULE POLINOM;
   PROCEDURE POL(X: REAL):REAL;
   VAR F,XP: REAL; I: REAL;
   BEGIN XP:=1.0; F:= 0.0;
      FOR I:= 0 TO 5 DO F:= F+XP*P[I]; XP:=XP*X END
      RETURN F
   END POL;
END POLINOM.
```

Рис. 2

```
DEFINITION MODULE SymSetHandling;
   FROM M2Compiler IMPORT Symbol;
   CONST SetLength = 4;
   TYPE Symset = ARRAY [ 0 .. SetLength ] OF BITSET;
   MACRO VAR s0, addopsys: Symset; i: CARDINAL;
   MACRO PROCEDURE Set1 (VAR s: SymSet; sy: Symbol);
   BEGIN INCL( s[ ORD(sy) DIV 16 ], ORD(sy) MOD 16)
   END Set1;
BEGIN FOR I := 0 SetLengt DO S[0] := () END;
      addopsys := s0; Set1(addopsys,plus);
      Set1(addopsys,minus); Set1(addopsys,orsy)
END SymSetHandling.
```

Рис. 3

Для генерации версий программ достаточно включить в модуль SYSTEM макропроцедуры для ввода-вывода в период компиляции. При этом версии будут фиксироваться символьным файлом (рис. 4).

```
DEFINITION MODULE M;
   FROM SYSTEM IMPORT MWriteString,MRead;
   MACRO VAR ArrSz:CARDINAL; C: CHAR;
BEGIN MWriteString('Большой массив?');MRead(C);
   IF C='Y' THEN ArrSz:=1000 ELSE ArrSz:=10 END
END M.
MODULE X;
   FROM M IMPORT ArrSz;
   VAR ARR: ARRAY[1..ArrSz] OF CARDINAL;      .....
END X.
```

Рис. 4