Библиотечные вступления и отдельная трансляция процедур в трансляторе с Алгола 68

А.Н.Терехов

1. Важнейшим условием создания больших программ и программных комплексов является возможность отдельной трансляции модулей. В простейшем случае такими модулями могут быть процедуры без глобальных идентификаторов с передачей информации только через параметры, однако при этом резко увеличивается объем программ и ухудшается их эффективность. Следующим шагом может быть использование не описанных в процедуре индикаторов вида, приоритета и операций, а также идентификаторов-констант, то есть таких объектов, описание которых не требует динамических действий, так как вся информация о них сосредотачивается в таблицах транслятора.

Наиболее удобным (однако и самым трудным в реализации) средством является возможность использования глобальных переменных. При этом кроме таблиц транслятора необходимо иметь некоторый начальный модуль, в котором исполняются описания переменных, некоторые инициализирующие действия и вызываются отдельно транслированные процедуры. При трансляции процедуры нужно уметь правильно обращаться с переменными, определенными в начальном модуле.

2. В данной реализации Алгола 68 на ЕС ЭВМ процедура представляется в виде совершенно самостоятельного объекта, содержащего всю необходимую информацию для исполнения процедуры. Это облегчает переход к вызову отдельно транслированной процедуры: достаточно в вызове или в вычислении процедурного значения использовать **V**-константу вместо **A**-константы (**A**-константа определяет адрес в данном модуле, **V**-константа дает информацию редактору связей для присоединения модуля).

Чтобы сообщить трансляторy, что некоторый идентификатор связывается с отдельно транслированной процедурой, нужно описать этот идентификатор следующий образом:

<формальный описатель> <идентификатор> = vcon <имя **V**-константы>

Например, proc (int, real) bool def – vcon VBABC

Если запись vcon <имя **V**-константы > заменить одним символом v, то в качестве имени имя **V**-константы берутся первые 8 литер идентификатора (дополненные, если необходимо, пробелами до 8 литер).

Аналогичная запись применяется в описаниях операций:

```
op (int, real) real + = vcon myplus,
op (real, real) bool less = v;
```

Для второй операции будет использоваться **V**-константа LESS.

Естественно, отдельно транслированная процедура может быть написана на любом языке, лишь бы были выполнены правила оформления процедур, принятые в данной реализации. Проще всего удовлетворить всем требованиям при использовании языка ассемблера; для процедур, написанных на других языках, легко подготовить соответствующие «переходные»

модули. Так как контроль осуществляется только при вызове, а не внутри процедуры, могут использоваться отдельно транслированные процедуры в тех случаях, когда одинаковые действия выполняются для различных видов параметров.

В Алголе 68 процедура может использоваться самостоятельно без предварительного связывания её с идентификатором. Чтобы не нарушать эту традицию, в данной реализации конструкция vcon доведена до уровня основы, но поскольку вид её неизвестен, её можно использовать только в сильных позициях (как skip, go to, nil).

Например, proc (ref real, char) void (vcon move) (a,b).

Конструкция vcon может выдавать значения любых видов (а не только процедурных), например, struct (int a, real b) c = v.

Опять-таки, конструкцию vcon удобно использовать для задания областей, доступ к которым различные идентификаторы осуществляют в соответствии с разными видами, например,

ref int a = vcon intchar, ref struct (char a,b) c = vcon intchar.

Здесь идентификаторы а и с именуют одну и ту же область памяти длиной 2 байта, причем её можно рассматривать и как целое число, и как структуру из двух литер.

3. При разработке большой программы или программного комплекса удобно выделить описания часто встречающихся объектов и включить эти описания в таблицы транслятора вместо повторения их в каждом модуле. Это не только экономит длину текста и время трансляции, но и имеет самостоятельную методологическую ценность. Нужную группу описаний может сформировать самый опытный член коллектива разработчиков комплекса, значительно уменьшается риск ошибки при внесении изменений, так как изменение делается в одном месте, а не во многих модулях. Таким способом можно формулировать специализированные языки для многих коллективов из непрофессиональных программистов, подготавливая достаточно представительные наборы описаний видов объектов и операций над ними.

Подготовленный набор описаний оформляется в виде программы на Алголе 68 и подается на вход транслятору, работающему в особом режиме – режиме построения библиотечного вступления. Этот режим отличается от обычного тем, что таблицы, построенные во время трансляции, не теряются после окончания трансляции, а записываются в заданную библиотеку в форме загрузочных модулей. Если указать эту библиотеку при трансляции процедур, транслятор будет использовать не стандартные таблицы, а построенные по библиотечному вступлению.

В общем случае до программы пользователя должна отработать программа библиотечного вступления, которая отводит память под переменные и выполняет инициализирующие действия. Таким образом, библиотечное вступление может иметь вид: (proc (int) int a = v; int b := 1;… print (a(b)); … proc void c = v; … c; …)

4. В библиотечном вступлении пользователь имеет дополнительные возможности по заданию «надежных» идентификаторов, определению упрощенных процедур и их использованию в фактических описателях.

Если в описании тождества перед формальным описателем стоит символ simp (например, simp proc void a=v, b=v), это означает, что процедуры с указанными идентификаторами (a,b)

можно вызывать не стандартным способом, а упрощенным (без заказа памяти, сохранения регистров и так далее).

В Алголе 68 фактический описатель, содержащий границы массивов, также играет роль процедуры, например

mode m = [m,n] int; … m a; … m b;

Запись vcon abc [,] int обозначает фактический описатель, границы которого вычисляет упрощенная процедура abc.

Для статических идентификаторов вида ref m гораздо чаще используется разыменованное значение вида m, чем адрес этого значения; данная реализация позволяет повторно использовать разыменованное значение такого идентификатора в регистре, а при присваивании идентификатору нового значения сбросить с регистра информацию о старом значении. Однако, если адрес значения был каким-либо образом размножен, например, неразыменованное значение идентификатора передано параметром или описан другой идентификатор, как ref m u = x, то уследить за всеми присваиваниями данному имени становится очень трудно. Потому узнаваемыми считаются только разыменованные значения идентификаторов, имя которых используется только при присваивании им нового значения. Такие идентификаторы мы называем надежными. Если статический идентификатор описывается в библиотечном вступлении, транслятор не может предугадать всех использований этих идентификаторов, поэтому автор вступления должен сам пометить символом safe все идентификаторы, разыменованные значения которых желательно иметь узнаваемыми и которые используются только как надежные. Например, safe int a,b := 0,c;

Здесь a,b,c объявляются надежными. Неправильное использование надежного идентификатора сигнализируется транслятором.

В разработке языковых средств и реализации программ построения библиотечных вступлений участвовали: С.Н.Баранов, И.Б.Гиндыш, Н.Р.Ноздрунов, А.Н.Терехов, Б.А.Федотов, Г.С.Цейтин, Г.А.Швецова.