Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Академии наук СССР

С.А. Романенко

СИСТЕМА ПРОГРАММИРОВАНИЯ РЕФАЛ – 2 ДЛЯ ЕС ЭВМ. КОМПИЛЯЦИЯ И ИСПОЛНЕНИЕ РЕФАЛ – ПРОГРАММ ПОД УПРАВЛЕНИЕМ ПДО СВМ

Москва 1987 г.

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В.Келдыша
Академии Наук СССР

С.А.Романенко

Система программирования Рефал-2 для ЕС ЭВМ. Компиляция и исполнение рефал-программ под управлением ПДО СВМ.

Москва
1987

УДК 681.3.06

В работе описано, как производить компиляцию и запуск программ, написанных на Рефале-2, под управлением ПДО СВМ. Кроме того, описаны средства отладки и сбора статистики для рефал-программ.

КЛЮЧЕВЫЕ СЛОВА И ФРАЗЫ: ЕС ЭВМ, обработка символьной информации, отладка программ, рекурсия, рефал, тестирование программ, функциональное программирование.

С О Д Е Р Ж А Н И Е

В В Е Д Е Н И Е

Компиляция, загрузка и исполнение рефал-программ [Р2ОВЯ 1987], [Р2ОБФ 1986] может производиться под управлением операционнной системы CMS или ПДО, которая, в свою очередь, работает в рамках VM или СВМ [СВМ 1985]. Предоставляемые средства отладки аналогичны тем, которые предлагались в предшествующих реализациях рефала [БР 1977], [КПРТР 1975], [КР 1975].

I. ГДЕ НАХОДИТСЯ РЕФАЛ-СИСТЕМА

Рефал-система представляет собой совокупность об'ектных модулей, командных файлов и символьной библиотеки, которые хранятся в следующих файлах.

REFAL EXEC - командный файл для запуска компилятора с рефала.

REFAL MODULE - компилятор с рефала в виде абсолютного образа памяти.

REFAL TXTLIB - библиотека об'ектных модулей, которая содержит первичные функции, а также программы, обеспечивающие интерфейс рефала и PL/I.

REFAL MACLIB - символьная библиотека, которая содержит разделы DCLREFAL, DCLTAG, DCLST и другие, которые могут включаться в программы на PL/I, взаимодействующие с рефал-программами с помощью макро-предложений %INCLUDE.

Перед компиляцией PL/I-программ, использующих библиотеку REFAL MACLIB, следует сделать ее доступной с помощью команды CMS/ПДО

```
GLOBAL MACLIB REFAL
```

Перед загрузкой рефал-программ следует сделать доступными библиотеки об'ектных модулей PLILIB TXTLIB и REFAL TXTLIB с помощью команды CMS/ПДО

```
GLOBAL TXTLIB REFAL PLILIB
```

При работе с рефал-системой, нужно обеспечить доступ к минидискам, на которых находятся PL/I-система и рефал-система. Для этого нужно воспользоваться командой CMS/ПДО ACCESS.

Пусть, например, PL/I-система находится на минидиске, который подключен к виртуальной машине по адресу 319, а рефал-система находится на одном из минидисков пользователя. Тогда перед началом работы с рефал-системой следует выполнить команду CMS/ПДО

```
ACCESS 319 P
```

Проще всего включить вышеупомянутые команды в файл PROFILE EXEC, на минидиске пользователя, с тем, чтобы они исполнялись автоматически всякий раз, когда загружается CMS/ПДО. Тогда файл PROFILE EXEC может иметь следующий вид:

```
&CONTROLE ERROR
ACCESS 319 P
GLOBAL TXTLIB REFAL PLILIB
GLOBAL MACLIB REFAL
```

2. ПОДГОТОВКА ИСХОДНЫХ ТЕКСТОВ РЕФАЛ-ПРОГРАММ

Исходные тексты рефал-программ можно подготавливать с помощью любого редактора текстов, например редактора XEDIT.

Файл, в котором содержатся исходные рефал-программы должен иметь записи переменной или фиксированной длины, длиной не более 80 литер.

Для того, чтобы редактор текстов автоматически устанавливал нужные атрибуты при создании файлов типа REFAL, необходимо предусмотреть это в файле PROFILE для редактора текстов.

При работе над данной реализацией рефала использовался редактор XEDIT, для которого был создан следующий файл PROFILE XEDIT.

```
&FT = &2
&I = &POSITION OF &FT REFAL PLIOPT SCRIPT ASSEMBLE EXEC
&IF &I ¬= 0 &GOTO -&FT
&EXIT
-REFAL
SET RECFM V
SET TRUNC 72
SET ZONE 1 71
SET TABS 1 10 18 72
SET SERIAL OFF
&EXIT
-PLIOPT
SET RECFM V
SET ZONE 1 72
SET TABS   2 11 14 17 20 23 26 29 32 35 38 41 44 47
SET SERIAL OFF
LINEND OFF
&EXIT
-SCRIPT
```

```
SET CASE MIXED RESPECT
SET RECFM V
SET LRECL 72
SET TRUNC 72
SET PREFIX ON RIGHT
SET ZONE 1 72
SET TABS 1 10 13 16 19 22 25 28 31 34 37 40 43 46
SET IMAGE ON
SET SERIAL OFF
LINEND OFF
&EXIT
-ASSEMBLE
SET SERIAL OFF
&EXIT
-EXEC
CASE MIXED
&EXIT
```

3.КОМПИЛЯЦИЯ РЕФАЛ-ПРОГРАММ

За один запуск рефал-компилятора можно скомпилировать один или несколько модулей. Модули, подлежащие компиляции, должны находиться в файле на одном из минидисков, доступных на чтение.

Чтобы скомпилировать модули, находящиеся в этом файле, необходимо выдать следующую команду CMS/ПДО:

REPAL F OPLIST

где F - идентификатор файла, а OPLIST - список опций.

Идентификатор файла F имеет вид: FN FT FM, где FN - имя файла, FT - тип файла, а FM - имя минидиска.

FM и FT могут быть опущены. Если опущено FM, то считается, что FM=A. Если опущены и FT и FM, то считается, что FT=REFAL, а FM=A.

Список опций OPLIST либо пуст, либо имеет вид:

(OPTION-1 OPTION-2 ... OPTION-K)

где OPTION-I имя опции, представляющее собой цепочку литер, не содержащую пробелов. Имена опций отделяются друг от друга одним или несколькими пробелами. Правая скобка ")" может быть опущена.

Список опций служит для установки режимов работы компилятора. Допускаются следующие имена опций.

NS - запретить печать исходного текста рефал-программы. Если не задана эта опция, исходный текст программы печатается.

NX - запретить печать таблицы перекрестных ссылок. Если не задана эта опция, таблица перекрестных ссылок печатается.

BESM - установить режим совместимости с БЭСМ-6. В этом режиме все идентификаторы, длиной более шести литер различаются по первым четырем и двум последним литерам, а все остальные литеры - игнорируются. Если не задана эта опция, идентификаторы различаются по первым 255 литерам, а все последующие литеры - игнорируются.

C - установить режим компиляции, обеспечивающий возможность собирать статистику о количестве применений каждого рефал-предложения в процессе исполнения рефал-программы.

FN - установить режим полных имен. Если эта опция

задана, то ко всем внутренним именам модуля спереди приписывается имя модуля, а вслед за ним - литера ":". Например, если MOD - имя модуля, а FUNC - имя функции, то все вхождения метки FUNC преобразуются в MOD:FUNC. Эти расширенные имена затем используются при исполнении программы для управления прокруткой и печати статистики.

PGT - установить режим печати заголовков страниц. Если задана эта опция, в начале каждой страницы печатается строчка из звездочек.

П Р И М Е Р 1.

```
REFAL  EXMPL
```

Компилируются модули из файла EXMPL REFAL A. Печатаются исходные тексты модулей и таблицы перекрестных ссылок.

П Р И М Е Р 2.

```
REFAL  EXMPL2 REFAL D (BESM NX
```

Компилируются модули из файла EXMPL2 REFAL D. Исходные тексты печатаются, а таблицы перекрестных ссылок - нет. Длинные идентификаторы укорачиваются до шести литер.

Если при компиляции не было обнаружено ни одной ошибки, компилятор вырабатывает код возврата RC=Ø, в противном случае RC=8.

Если при компиляции файла FN FT FM не было обнаружено ни одной ошибки ни в одном из модулей, то в конце компиляции оказываются порожденными два новых файла: файл с листингом и файл, содержащий результат компиляции - один или несколько об'ектных модулей. Эти файлы помещаются на минидиск FR, который определяется следующим образом.

Если минидиск FM доступен на запись, то FR=FM. Иначе, FR - это первый из минидисков, доступных на запись, если рассматривать их в том порядке, в котором CMS/ПДО производит поиск файлов на минидисках.

Листинг записывается в файл

```
FN LISTING FR
```

а об'ектный модуль - в файл

```
FN TEXT    FR
```

Кроме того, на терминал выводится протокол работы компилятора, который содержит имена компилируемых модулей и сообщения об ошибках, если таковые обнаруживаются в процессе компиляции.

Файл FN LISTING FR имеет атрибуты: DDNAME=SYSPRINT, RECFM=V, LRECL=121.

Файл FN TEXT FR имеет атрибуты: DDNAME=SYSLIN, RECFM=F, LRECL=80.

Если в процессе компиляции обнаруживаются ошибки в исходных рефал-программах, то после завершения компиляции файл FN TEXT FR уничтожается, а код возврата RC=8.

Кроме вышеупомянутых, во время работы компилятора создаются еще два временных файла:

```
FILE  SYSUT1  FU
FILE  SYSUT2  FU
```

Эти файлы размещаются на том минидиске FU из числа доступных на запись, на котором перед началом работы компилятора имеется наибольшее свободное пространство. В конце работы компилятора эти файлы уничтожаются.

Файл FILE SYSUT1 FU имеет следующие атрибуты: DDNAME=SYSUT1, RECFM=F, LRECL=80.

Файл FILE SYSUT2 FU имеет следующие атрибуты: DDNAME=SYSUT2, RECFM=F, LRECL=12.

4. ИСПОЛНЕНИЕ РЕФАЛ-ПРОГРАММ.

Исполнение готовой рефал-программы производится с помощью PL/I-подпрограммы RFEXEC. Эта подпрограмма имеет один параметр - входную точку некоторого рефал-модуля, которая является именем некоторой функции FUNC. Обращение к RFEXEC из PL/I-программы имеет вид:

```
CALL RFEXEC(FUNC);
```

При этом RFEXEC и FUNC должны быть об'явлены следующим образом:

```
DECLARE RFEXEC   ENTRY(ENTRY());
DECLARE FUNC     EXTERNAL ENTRY();
```

В начале работы RFEXEC формирует начальное поле зрения вида

```
<FUNC>
```

и запускает "рефал-машину".

Когда работа рефал-программы завершается, RFEXEC печатает количество выполненных шагов, причину остановки "рефал-машины", а также содержимое поля зрения и копилки, если они не пусты.

Допустим, что требуется запустить рефал-программу начиная

с входной точки JOB. Тогда простейшая PL/I-программа, запускающая рефал-программу имеет вид:

```
EXECJOB: PROCEDURE OPTIONS(MAIN);
         DECLARE  RFEXEC   ENTRY(ENTRY());
         DECLARE  JOB      EXTERNAL ENTRY();
         CALL RFEXEC(JOB);
         END EXECJOB;
```

Эту программу следует поместить в файл EXECJOB PLIOPT и скомпилировать с помощью команды CMS/ПДО:

```
PLIOPT EXECJOB (MAR(1,72) SEQ(73,80) OPT(TIME)
```

Затем следует загрузить и выполнить эту программу следующим образом.

Пусть файл JOB TEXT содержит загружаемые рефал-модули, в одном из которых имеется входная точка JOB. Тогда загрузка и запуск могут быть сделаны с помощью следующих команд CMS/ПДО:

```
GLOBAL TXTLIB REFAL PLILIB
LOAD   EXECJOB JOB  (NODUP NOMAP
FILEDEF * CLEAR
FILEDEF SYSIN     ...
FILEDEF SYSPRINT ...
START   * ISASIZE(10K)
```

Входной файл SYSIN и выходной файл SYSPRINT следует определить предложениями FILEDEF. Например, если SYSIN назначен на терминал, а SYSPRINT - на виртуальный принтер, следует задать

```
FILEDEF SYSIN     TERMINAL
FILEDEF SYSPRINT  PRINTER
```

Если же SYSIN назначается на файл XXX YYY на минидиске D, а

файл SYSPRINT назначается на файл PP QQ на минидиске A, следует задать:

```
FILEDEF SYSIN    DISK XXX YYY D
FILEDEF SYSPRINT DISK PP  QQ  A
```

Файл SYSPRINT нужен всегда, файл SYSIN требуется в том случае, если рефал-программа читает данные с помощью первичной функции CARD.

Если SYSIN и SYSPRINT не описаны явно командами FILEDEF, то по умолчанию они назначаются на терминал.

Все тексты, которые выводятся в файл SYSPRINT с помощью первичных функций PRINT и PROUT, разбиваются на печатные строки длиной в I20 литер. Однако, длину печатной строки можно увеличить. Для этого достаточно описать файл SYSPRINT в головной программе EXECJOB и открыть его явно, задав атрибут LINESIZE.

Например, если требуется установить длину строки в I28 литер, следует переделать EXECJOB следующим образом:

```
EXECJOB: PROCEDURE OPTIONS(MAIN);
         DECLARE
            JOB      ENTRY() EXTERNAL,
            RFEXEC   ENTRY(ENTRY());
         DECLARE
            SYSPRINT EXTERNAL FILE PRINT STREAM;
         OPEN FILE(SYSPRINT) LINESIZE(128);
         CALL RFEXEC(JOB);
         END EXECJOB;
```

5. СРЕДСТВА ОТЛАДКИ

5.I. ВЫЗОВ ПОДПРОГРАММЫ ОТЛАДКИ

Подпрограмма RFEXEC не содержит средств отладки. Ею следует пользоваться для запуска уже отлаженных рефал-программ.

При отладке программ следует пользоваться подпрограммой RFDBG.

Обращение к RFDBG ничем не отличается от обращения к RFEXEC, поэтому простейшая PL/I-программа, запускающая RFDBG, имеет вид:

```
DBGJOB:  PROCEDURE OPTIONS(MAIN);
         DECLARE
            JOB      ENTRY() EXTERNAL,
            RFDBG    ENTRY(ENTRY());
         CALL RFDBG(JOB);
         END DBGJOB;
```

При вызове этой программы следует определить предложениями FILEDEF те же файлы, которые требовались при запуске с помощью подпрограммы RFEXEC. Помимо этих файлов следует определить файл SYSDBG. Этот файл имеет атрибуты DDNAME=SYSDBG, RECFM=F, LRECL=8Ø. Он содержит предложения, управляющие работой подпрограммы RFDBG.

RFDBG начинает свою работу с того, что она читает и анализирует управляющие предложения. Если при этом обнаруживаются ошибки в управляющих предложениях, работа RFDBG на этом заканчивается. Если же в управляющих предложениях ошибок нет, начинается исполнение рефал-программы.

Любое из следующих предложений FILEDEF может служить описанием для файла SYSDBG:

```
FILEDEF SYSDBG  DISK FILE25 SYSDBG A
FILEDEF SYSDBG  TERMINAL
FILEDEF SYSDBG  DUMMY
```

Первое предложение назначает SYSDBG на файл с идентификатором FILE25 SYSDBG, расположенный на минидиске A. Этот файл должен быть предварительно создан, например, редактором текстов. Второе предложение назначает SYSDBG на терминал. В этом случае признаком конца файла служит пустая входная строка, т.е. простое нажатие на клавишу "ВВОД". Третье предложение назначает SYSDBG на фиктивный файл, что равносильно назначению SYSDBG на пустой файл.

Если при запуске RFDBG не требуется задавать какие-либо управляющие предложения, можно вообще не определять файл SYSDBG. В этом случае RFDBG распознает, что файл SYSDBG не определен и работает так, словно SYSDBG пуст.

Каждая запись в файле SYSDBG содержит либо комментарий, либо признак конца управляющих предложений, либо управляющее предложение.

Любая запись, которая содержит литеру "*" в первой позиции, является комментарием и не оказывает никакого влияния на работу программы RFDBG. Таким образом, в такой записи начиная со второй позиции можно помещать произвольную информацию.

Если запись содержит "/*␣" в первых трех позициях, она является признаком конца управляющих предложений. Прочитав эту запись RFDBG прекращает дальнейшее чтение файла SYSDBG и закрывает его. Присутствие такой записи в файле SYSDBG не обязательно. Если ее нет, RFDBG прочитывает весь файл SYSDBG до конца.

Запись типа "/*␣" полезна в тех случаях, когда загрузка и запуск рефал-программы выполняются с помощью файла типа EXEC. Тогда, вместо того, чтобы заводить отдельный файл,

содержащий управляющие предложения, можно поместить эти предложения непосредственно в файл типа EXEC следующим образом:

```
&ERROR &EXIT &RETCODE
GLOBAL TXTLIB REFAL PLILIB
LOAD   DBGJOB JOB  (NODUP NOMAP
FILEDEF * CLEAR
FILEDEF SYSDBG TERMINAL
FILEDEF SYSIN    ...
FILEDEF SYSPRINT ...
&BEGSTACK
*КОММЕНТАРИЙ: ДАЛЬШЕ РАСПОЛОЖЕНЫ УПРАВЛЯЮЩИЕ
* ПРЕДЛОЖЕНИЯ, ПРИЗНАКОМ КОНЦА КОТОРЫХ ЯВЛЯЕТСЯ /*
MEMORY
¬=
STOP  2ØØ
/*
&END
START   * ISASIZE(1ØK)
```

При желании можно поместить в файл типа EXEC не только содержимое файла SYSDBG, но и содержимое файла SYSIN. Для этого достаточно назначить файл SYSIN на терминал, а его содержимое поместить сразу же вслед за содержимым файла SYSDBG. Признаком конца файла SYSIN служит запись, которая начинается с "/*␣". Таким образом, EXEC-файл должен иметь следующий вид:

```
&ERROR &EXIT &RETCODE
GLOBAL TXTLIB REFAL PLILIB
LOAD   DBGJOB JOB  (NODUP NOMAP
FILEDEF * CLEAR
FILEDEF SYSDBG TERMINAL
FILEDEF SYSIN  TERMINAL
FILEDEF SYSPRINT ...
&BEGSTACK
*КОММЕНТАРИЙ: ДАЛЬШЕ РАСПОЛОЖЕНЫ УПРАВЛЯЮЩИЕ
```

```
* ПРЕДЛОЖЕНИЯ, ПРИЗНАКОМ КОНЦА КОТОРЫХ ЯВЛЯЕТСЯ /*
* ЗАТЕМ РАСПОЛОЖЕНО СОДЕРЖИМОЕ ФАЙЛА SYSIN,
* ПРИЗНАКОМ КОНЦА КОТОРОГО ЯВЛЯЕТСЯ /*
MEMORY
¬=
STOP  2ØØ
/*
1111
2222 ТРИ ЗАПИСИ ДЛЯ SYSIN
3333
/*
&END
START   * ISASIZE(1ØK)
```

5.2. УПРАВЛЯЮЩИЕ ПРЕДЛОЖЕНИЯ

Все управляющие предложения имеют один из следующих трех форматов:

```
KEY
KEY  N
KEY  F1 F2 ... FN
```

где KEY - ключевое слово, которое представляет собой последовательность литер, не содержащую пробелов, и за которым должен следовать один или несколько пробелов, N - целое положительное число, F1 F2 ... FN - имена рефал-функций, которые разделяются между собой одним или несколькими пробелами. Список функций F1 F2 ... FN может быть пустым.

В качестве имени рефал-функции можно употреблять также литеру "%". Такое имя изображает множество всех имен символов-ссылок. Таким образом, все символы-ссылки с точки зрения управления прокруткой являются как бы одной функцией.

Каждое управляющее предложение занимает одну запись и располагается в позициях с I-й по 72-ю. Информация, находящаяся в позициях с 73 по 80, игнорируется и может использоваться для нумерации записей. Ключевое слово KEY должно начинаться с I-й позиции.

Например:

```
MEMORY
>= FUNCI PSI EPSILON
>=
STOP  IØØ
```

Все управляющие предложения подразделяются на следующие группы:

- предложения управления памятью;
- предложения управления остановом;
- предложения, управления прокруткой.

5.3. ПРЕДЛОЖЕНИЯ УПРАВЛЕНИЯ ПАМЯТЬЮ

В процессе работы рефал-программы происходит динамический захват памяти под поле зрения, копилку и ящики.

Эта память имеет списковую организацию и состоит из двенадцатибайтовых з в е н ь е в. Одно звено может содержать символ-литеру или составной символ, либо одну из скобок "(", ")", "<", ">".

Алгоритмом захвата памяти можно управлять, задавая пять параметров: MINSZ, LIMSZ, INCRSZ, COLLSZ и REMSZ.

Эти параметры имеют следующий смысл (все размеры задаются в двенадцатибайтовых звеньях).

MINSZ. Минимальный размер списка. В начале работы рефал-программы сразу захватывается под список MINSZ звеньев.

LIMSZ. Ограничение на максимальный размер списка. В процессе работы рефал-программы не разрешается тратить под список более чем LIMSZ звеньев, даже если еще имеется свободная память.

INCRSZ. Количество звеньев, которое добавляется к списку при каждом захвате дополнительной памяти.

COLLSZ. Количество звеньев, которое должно освобождаться после очередной сборки мусора. Если освободилось меньше, чем COLLSZ звеньев, это означает, что к списку следует добавить INCRSZ дополнительных звеньев.

REMSZ. Размер памяти (в звеньях), которая должна остаться не захваченной под список и, таким образом, может использоваться для нужд других программ и операционной системы (например, для создания буферов ввода-вывода и выполнения операций по открытию и закрытию файлов).

Значения этих параметров устанавливаются с помощью управляющих предложений

```
MINSZ   N
LIMSZ   N
INCRSZ  N
COLLSZ  N
REMSZ   N
```

Если значения каких-либо из этих параметров не заданы явно, то они по умолчанию получают следующие значения.

```
MINSZ       500
LIMSZ    2000000
INCRSZ      200
COLLSZ      200
REMSZ      1000
```

Можно получить подробную информацию о работе алгоритма распределения памяти в процессе выполнения рефал-программы. Для этого следует поместить среди управляющей информации предложение

```
MEMORY
```

Если это предложение задано, то перед началом работы рефал-программы будет напечатана следующая информация:

- используемые значения параметров MINSZ, LIMSZ, INCRSZ, COLLSZ, REMSZ;

- значения AVAILSZ (размер памяти, доступной для отведения под список) и MAXSZ (максимальный возможный размер списка с учетом AVAILSZ, LIMSZ и REMSZ).

Затем, во время работы рефал-программы, при каждом захвате дополнительной памяти печатается номер шага и количество захваченных звеньев, а при каждой сборке мусора - номер шага и количество освободившихся звеньев.

Например, если заданы предложения

```
MEMORY
LIMSZ  3000
```

то будет печататься информация о работе алгоритма управления памятью, а если размер поля зрения превысит 3000 звеньев, будет об'явлено, что "свободная память исчерпана".

5.4. ПРЕДЛОЖЕНИЯ УПРАВЛЕНИЯ ОСТАНОВОМ.

Имеются предложения управления остановом двух типов: STOP-предложения, которые позволяют задавать останов по номеру шага, и TRAP-предложения, которые позволяют задавать останов по имени функции.

Если задано предложение

```
STOP  N
```

то работа рефал-программы остановится, как только будет выполнено N шагов. Например, если задано

```
STOP  3ØØØØ
```

то рефал-программа остановится перед началом выполнения шага 3ØØØI.

Эта возможность полезна в тех случаях, когда отлаживаемая рефал-программа зацикливается.

Если задано предложение вида

```
TRAP  FI F2 ... FN
```

где FI F2 ... FN - список имен функций, это означает, что функции FI F2 ведущим становится функциональный терм вида

```
<F  [E]>
```

где F - функция, об'явленная ловушкой, а [E] - об'ектное выражение, рефал-программа останавливается, не приступая к выполнению очередного шага.

Эта возможность полезна в тех случаях, когда требуется

распечатать состояние поля зрения, копилки и ящиков в момент обращения к одной из функций-ловушек.

Функция считается ловушкой, если ее имя задано хотя бы в одном из TRAP-предложений. При этом под именем функции подразумевается то имя, которое она имеет внутри того модуля, в котором она описана.

Если TRAP-предложения заданы вместе со STOP-предложением, останов произойдет как только будет удовлетворено хотя бы одно из условий, заданных этими предложениями.

5.5. ПРЕДЛОЖЕНИЯ УПРАВЛЕНИЯ ПРОКРУТКОЙ

Средства отладки, предоставляемые подпрограммой RFDBG, сводятся к тому, что можно заказать "прокрутку" - печать протокола о ходе выполнения рефал-программы.

Прокрутка дает возможность получать информацию о выполнении отдельных шагов рефал-машины, а также об обращениях к указанным функциям и о результатах вычисления этих обращений.

При печати информации об отдельном шаге выдаются

- номер шага;
- ведущий функциональный терм;
- результат замены (результат выполнения шага).

При печати информации об обращениях к указанным функциям выдается

- номер шага, на котором произошло обращение;

- ведущий терм;

- номер шага, на котором завершилось полное вычисление обращения к функции;

- окончательный результат замены, т.е. то выражение, которое получается, когда в выражении, возникшем из обращения к функции, не осталось ни одного функционального терма.

Задание на прокрутку состоит из двух частей:

- диапазон прокрутки;

- условия прокрутки.

Диапазон прокрутки задается с помощью предложений

```
FROM  NFROM
TO    NTO
```

где NFROM и NTO – целые положительные числа.

Эти предложения означают, что следует печатать информацию только о тех шагах, номера которых лежат в интервале от NFROM до NTO. Таким образом, печать информации о тех шагах, номер которых меньше NFROM или больше NTO – запрещается. В частности, если NFROM>NTO, то никакая информация печататься не будет.

Если не заданы ни диапазон прокрутки, ни условия прокрутки, то считается, что прокрутка не нужна, и никакая информация не печатается.

Если предложение "FROM" не задано, но задано предложение "TO" или хотя бы одно условие прокрутки, то считается, что NFROM=I.

Если же предложение "TO" не задано, но задано предложение "FROM" или хотя бы одно условие прокрутки, то считается, что NTO=2I47483647, т.е. практически "бесконечности".

Таким образом, получается, что если диапазон прокрутки не задан, но задано хотя бы одно условие прокрутки, то NFROM=I и NTO=2I47483647.

Если диапазон прокрутки не пуст, и не задано ни одного условия прокрутки, то считается, что следует печатать информацию обо всех шагах, лежащих в этом диапазоне.

Приведем примеры заданий на прокрутку, состоящие только из указания диапазона прокрутки.

П Р И М Е Р I.

```
FROM I
```

Печатается информация обо всех шагах.

П Р И М Е Р 2.

```
FROM I000
```

Печатается информация обо всех шагах, начиная с шага I000.

П Р И М Е Р 3.

```
TO 5000
```

Печатается информация обо всех шагах, начиная с шага I до шага 5000 включительно.

П Р И М Е Р 4.

```
FROM  5ØØ
TO    6ØØ
```

Печатается информация обо всех шагах, начиная с шага 500 до шага 600 включительно.

Теперь рассмотрим, как задается вторая часть задания на прокрутку - условия прокрутки.

В то время как диапазон прокрутки накладывает ограничения на номера шагов, для которых должна печататься информация, условия прокрутки указывают, какую именно информацию и о каких функциях следует печатать.

Условия прокрутки задаются следующим образом.

Считается, что с каждой функцией связано шесть условий прокрутки: ">" , ">=", "=", "⁻=", "<" , "<=", каждое из которых представляет собой бит, который может находиться в одном из двух состояний "Ø" или "1". Если этот бит равен "1", будем говорить, что соответствующее условие для функции задано, а если равен "Ø" - будем говорить, что соответствующее условие не задано.

Каждое условие может задаваться независимо от всех других с помощью предложения вида

```
KEY  F1 F2 ... FN
```

где KEY совпадает с именем соответствующего условия, а F1 F2 ... FN - список имен функций, для которых задается это условие. Список функций, в частности, может быть пустым. Под именем функции подразумевается то имя, которое она имеет внутри того модуля, в котором она описана.

Например:

```
> FUNC1 FUNC2
>=  FUNC
= PSI FI
¬= FI EPS
< XXX
<= ZZZ
```

Приняты следующие дополнительные соглашения:

- если не задано н и о д н о г о условия с ключевым словом ">" или ">=", то считается, что ">=" задано для всех функций;

- если не задано н и о д н о г о условия с ключевым словом "=", то считается, что "=" задано для всех функций.

Условия прокрутки следующим образом истолковываются в процессе исполнения рефал-программы.

Управление прокруткой может находиться в одном из трех состояний: S1, S2 или S3. Оно может переходить из состояния S1 в состояние S2 и обратно, а также из состояния S2 в состояние S3 и обратно. Непосредственный переход из S1 в S3 или из S3 в S1 - невозможен.

Переходами между состояниями S1 и S2 управляют условия ">" и ">=". Переходами между состояниями S2 и S3 управляют условия "<" и "<=". Условия "=" и "¬=" правляют печатью информации в состоянии S2. Таким образом, получается схема переходов, изображенная на Рис.1.

В начале работы прокрутка находится в состоянии S1.

Находясь в состоянии S1, прокрутка выполняет рефал-программу до тех пор, пока не встретится обращение к функции, для которой было задано одно из условий ">" или ">=". Тогда прокрутка переходит в состояние S2, причем, если задано

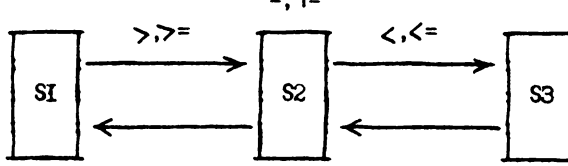

Рис.I. Схема переходов между состояниями прокрутки.

">=", но не задано ">", то печатается обращение к этой функции.

Возврат в состояние S1 происходит только после того, как полностью закончится вычисление обращения к функции, по которому произошел переход в S2 из S1. При этом, если для этой функции было задано ">=", но не задано ">", то при возврате в S1 печатается окончательный результат замены, т.е. то выражение, которое возникло в результате полного вычисления обращения к функции.

Находясь в состоянии S2, прокрутка выполняет рефал-программу до тех пор, пока не встретится обращение к функции, для которой было задано условие "<" или "<=". Тогда прокрутка переходит в состояние S3, причем, если задано "<=", но не задано "<", то печатается обращение к этой функции.

Возврат в состояние S2 из состояния S3 происходит только после того, как полностью закончится вычисление обращения к функции, по которому произошел переход в S3 из S2. При этом, если для этой функции было задано "<=", но не задано "<", то при возврате в S2 печатается окончательный результат замены, т.е. то выражение, которое возникло в результате полного вычисления обращения к функции.

Если прокрутка находится в состоянии S2, и для текущей функции не задано ни "<", ни "<=", то прокрутка выполняет один шаг и остается в состоянии S2. При этом, если для текущей функции задано условие "=", но не задано условие

"¬=", то печатается обращение к функции и результат непосредственной замены (результат выполнения шага).

Например, если заданы условия

```
>=  X
=   F1 F2 F3
<=  Z1 Z2
```

то печать будет начинаться при каждом обращении к функции X. Будет печататься обращение к X и результат ее полного вычисления. В процессе вычисления X будут печататься обращения к F1, F2, F3 и непосредственные результаты замены, а также обращения к Z1 и Z2 и результаты полного вычисления этих обращений.

Иногда требуется, чтобы печатались только обращения к указанным функциям и окончательные результаты замены. Этого можно добиться задав условие "=" с пустым списком функций. Тогда будет считаться, что "=" задано для пустого множества функций, что приведет к тому, что будут печататься обращения только к тем функциям, для которых явно указано условие "<=".

Например, если задать условия

```
=
<=  FUNC1 FUNC2
```

то будут печататься только обращения к FUNC1 и FUNC2 и полные результаты замены.

И, наконец, в тех случаях, когда требуется запустить прокрутку по всем функциям, с первого шага до последнего, можно задать условие "¬=" для пустого множества функций. Таким образом, задание на прокрутку будет иметь следующий вид:

7=

5.6. ПЕРЕХВАТ ОСТАНОВА ПО ОТОЖДЕСТВЛЕНИЮ

Легкость отладки рефал-программ в значительной степени обусловлена тем, что различные нарушения в структуре обрабатываемых об'ектов, как правило, довольно быстро приводят к авосту "отождествление невозможно". Однако это приятное свойство утрачивается, если какие-то функции, написанные на PL/I (или языке ассемблера), не проверяют правильность аргумента. Поэтому при отладке функций, написанных на PL/I, необходимо тщательно тестировать как те случаи, когда обращение к функции имеет допустимый вид, так и те случаи, когда аргумент функции задан неверно. Всегда, когда аргумент не принадлежит к области определения функции, эта функция должна вырабатывать авост "отождествление невозможно".

Таким образом, всякий тест, предназначенный для проверки функций, написанных на PL/I, должен включать как правильные, так и заведомо неправильные обращения к этим функциям.

Для прогона таких тестов предусмотрен особый режим "EI=". Если рефал-программа исполняется в этом режиме, то при возникновении состояния "отождествление невозможно" выполнение рефал-программы не прекращается. Вместо этого печатается номер текущего шага, предупреждающее сообщение и ведущий функциональный терм. Затем ведущий терм заменяется на пустое выражение и работа рефал-программы продолжается.

Таким образом, рефал-программа исполняется так, словно в конце каждой функции добавлено предложение

```
EI =
```

Этому обстоятельству режим "EI=" и обязан своим

названием.

Для того, чтобы рефал-программа исполнялась в режиме "EI=", необходимо задать управляющее предложение

EI=

5.7. ВЫБОР ФОРМАТА ПЕЧАТИ СОСТАВНЫХ СИМВОЛОВ

Выражения, которые употребляются в рефал-программах, могут содержать и составные символы, и переменные. Поэтому, чтобы как-то отличать их друг от друга, возникает необходимость пометить либо составные символы, либо скобки. Поскольку в рефал-программах, как правило, переменные встречаются значительно чаще, чем составные символы, помечаются именно символы. А именно, составные символы обрамляются знаками "/".

Во время прокрутки печатаются только выражения, не содержащие переменных, поэтому не возникает проблемы различения составных символов и переменных. Следовательно, знаки "/", в которые заключены составные символы, не несут никакой информации и их можно было бы не печатать.

Чтобы отменить печать знаков "/", обрамляющих составные символы, достаточно задать предложение

NOSLASH

6. СРЕДСТВА СБОРА СТАТИСТИКИ

Подпрограмма RFDBG не содержит средств сбора статистики о количестве обращений к различным функциям и предложениям рефал-программы, а также о времени центрального процессора, затраченном на исполнение отдельных функций. Все эти средства содержатся в подпрограмме RFCNT, которая вызывается точно так же, как RFDBG и, помимо средств сбора статистики, обеспечивает все возможности по отладке программ, которые имеются в RFDBG.

Возникает вопрос: зачем же нужна подпрограмма RFDBG, когда вместо нее всегда можно использовать RFCNT? Ответ состоит в том, что RFDBG занимает меньше места в памяти, поэтому ее можно использовать в тех случаях, когда сбор статистики не нужен, а памяти не хватает.

Если вместо RFDBG используется RFCNT, то в файле SYSDBG разрешается использовать следующие дополнительные предложения:

```
FCOUNT
SCOUNT
COUNT
PTIMER
```

Если задано предложение FCOUNT, это означает, что следует собирать статистику об обращениях к рефал-функциям.

Если задано предложение SCOUNT, это означает, что следует собирать статистику о применениях отдельных предложений в рефал-функциях.

Если задано предложение PTIMER, это означает, что следует собирать статистику о времени центрального процессора, затраченном на исполнение каждой из рефал-функций.

Предложение COUNT эквивалентно двум предложениям: FCOUNT и SCOUNT.

Если задано предложение PTIMER, то собирается статистика не только о времени работы функций, но и о количестве обращений к ним, поэтому в этом случае предложение FCOUNT можно не задавать.

Сведения о работе рефал-программы, собранные во время ее работы, выводятся в файл SYSPRINT после завершения работы рефал-программы.

Если были заданы режимы FCOUNT или PTIMER, то печатается таблица, каждая строка которой содержит следующую информацию:

- имя функции;

- общее количество вызовов этой функции;

- общее количество вызовов этой функции в процентах, по отношению к числу всех вызовов (которое совпадает с числом завершенных шагов);

- время центрального процессора (в микросекундах), затраченное на работу функции;

- время центрального процессора, затраченное на работу функции, в процентах, по отношению к общему времени, затраченному на работу всех функций;

- средняя продолжительность шага (в микросекундах), которая получается если разделить время, затраченное на работу функции, на количество обращений к этой функции.

Если был задан режим SCOUNT, то печатается таблица, в каждой строке которой содержится следующая информация:

- имя функции;

- сведения о количестве применений каждого из предложений.

Сведения о количестве применений каждого предложения печатаются в виде последовательности чисел. Первое число - это количество применений первого предложения, второе число - количество применений второго предложения и т.д. Если функция содержит более десяти предложений, то эта последовательность чисел переносится на следующие строки. При этом, в каждой строке печатаются сведения о десяти очередных предложениях.

Если какое-то предложение не применялось ни разу, то (вместо числа 0) в соответствующей позиции печатается пять звездочек. Очевидно, что в процессе тестирования каждое предложение следует проверить хотя бы один раз, поэтому звездочки - это предупреждение о том, что тест заведомо не обладает достаточной полнотой.

Следует обратить внимание на следующие обстоятельства.

- Сведения о функциях печатаются в том порядке, в котором функции расположены в памяти после загрузки. Взаимное расположение функций внутри одного модуля - такое же, как в исходной рефал-программе. Взаимное расположение модулей определяется в результате загрузки.

- Если в разных модулях используются функции с одинаковыми именами, можно при компиляции этих модулей задать опцию "FN". В результате этого в имя каждой функции будет вставлено имя соответствующего модуля.

- В таблицах печатаются сведения только о тех функциях, к которым было хотя бы одно обращение, поэтому обнаружить функции, к которым не было ни одного

обращения, можно только "вручную", сличая текст рефал-программы с таблицей.

- Статистика об отдельных предложениях будет собрана только в том случае, если при компиляции соответствующих рефал-модулей была задана опция "С".

И, наконец, несколько важных обстоятельств следует учитывать при сборе статистики о временах работы функций.

Если задано предложение PTIMER, то для измерения времени используется таймер центрального процессора. Во-первых, этот таймер имеется не на всех моделях ЕС ЭВМ. Во-вторых, для работы с ним RFCNT использует привилегированные команды SPT и STPT, поэтому при исполнении программы должен быть установлен режим расширенного управления.

Таким образом, если рефал-программа работает под управлением ОС ЕС, задавать режим PTIMER нельзя. Если же рефал-программа должна работать под управлением CMS/ПДО, то, прежде чем загружать ее, следует установить для виртуальной машины режим расширенного управления и перезагрузить CMS/ПДО. Для этого нужно нажать клавишу "ПДП", а затем выдать команды

```
SET ECMODE ON
İ CMS
```

ЛИТЕРАТУРА

[БР 1977]

Базисный рефал и его реализация на вычислительных машинах. М.: ЦНИПИАСС, 1977.

[КПРТР 1975]

Ан.В.Климов, Л.В.Проворов, С.А.Романенко, Е.В.Травкина. Рефал в мониторной системе "Дубна" БЭСМ-6. Входной язык компилятора и запуск программ. Препринт ИПМ АН СССР N 8, М., 1975.

[КР 1975]

Ан.В.Климов, С.А.Романенко. Рефал в мониторной системе "Дубна" БЭСМ-6. Интерфейс рефала и фортрана. ИПМ АН СССР, М., 1975.

[Р2ОБФ 1986]

Анд.В.Климов, С.А.Романенко. Система программирования Рефал-2 для ЕС ЭВМ. Описание библиотеки функций. Препринт ИПМ им.М.В.Келдыша АН СССР N 200, М.,1986.

[Р2ОВЯ 1987]

Анд.В.Климов, С.А.Романенко. Система программирования Рефал-2 для ЕС ЭВМ. Описание входного языка. ИПМ АН СССР, М., 1987.

[СВМ 1985]

Система виртуальных машин для ЕС ЭВМ. Справочник / Под ред. Э.В.Ковалевича. - М.: Финансы и статистика, 1985.

[ФО 1983]

Г.Д.Фролов, В.Ю.Олонин. Практический курс программирования на языке PL/I. - М.: Наука. Главная редакция физико-математической литературы, 1983.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ УПРАВЛЯЮЩИХ ПРЕДЛОЖЕНИЙ

С.А. Романенко " Система программирования Рефал – 2 для ЕС ЭВМ
Компиляция и исполнение рефал – программ под управлением ПДО СВМ.
Редактор А.С. Штаркман. Корректор А.В. Климов.

Подписано к печати 17.04.87г. № Т-10034. Заказ № 194.
Формат бумаги 60X90 1/16. Тираж 355 экз.
Объём 1,7 уч.-изд.л. Цена 20 коп.

055 (02)2

Отпечатано на ротапринтах в Институте прикладной математики АН СССР
Москва, Миусская пл. 4.

©

Цена 20 коп.