ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 200 за 1986 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

А. В. Климов, С. А. Романенко

Система программирования РЕФАЛ-2 для ЕС ЭВМ. Описание библиотеки функций

Рекомендуемая форма библиографической ссылки: Климов А. В., Романенко С. А. Система программирования РЕФАЛ-2 для ЕС ЭВМ. Описание библиотеки функций // Препринты ИПМ им. М.В.Келдыша. 1986. № 200. 42 с.
https://library.keldysh.ru/preprint.asp?id=1986-200

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

А.В. Климов, С.А. Романенко

СИСТЕМА ПРОГРАММИРОВАНИЯ РЕФАЛ-2 ДЛЯ ЕС ЭВМ

ОПИСАНИЕ БИБЛИОТЕКИ ФУНКЦИЙ

Препринт № 200 за 1986г.

Москва

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

Анд.В.Климов, С.А.Романенко

СИСТЕМА ПРОГРАММИРОВАНИЯ РЕФАЛ-2 ДЛЯ ЕС ЭВМ. ОПИСАНИЕ БИБЛИОТЕКИ ФУНКЦИЙ.

Москва
1986

УДК 681.3.06

Описана библиотека функций, поставляемая с системой программирования рефал-2 для ЕС ЭВМ. Язык рефал (алгоритмический язык рекурсивных функций) предназначен для обработки символьной информации, представленной в виде выражений, имеющих древовидную структуру. Основными изобразительными средствами рефала являются сопоставление с образцом, подстановка и рекурсия. Библиотека содержит функции ввода-вывода текстов в последовательные файлы, арифметических операций над целыми числами неограниченной разрядности и ряд других функций.

КЛЮЧЕВЫЕ СЛОВА И ФРАЗЫ: рефал, обработка символьной информации, функциональное программирование, язык программирования, библиотека подпрограмм.

С О Д Е Р Ж А Н И Е

ВВЕДЕНИЕ

В состав системы программирования рефал-2 входит набор функций, которые могут быть использованы в рефал-программах в качестве внешних и хранятся в библиотеке об'ектных модулей, поставляемых с рефал-системой.

Чтобы использовать любую из этих функций в программе, написанной на рефале, достаточно об'явить ее внешней в соответствующем модуле с помощью директивы EXTRN .

Часть библиотечных функций написана на рефале и затем скомпилирована в об'ектные модули с помощью рефал-компилятора. Остальные функции запрограммированы на языке PL/I или на языке ассемблера. Эти функции именуются п е р в и ч н ы м и.

С точки зрения пользователя рефал-системы единственное отличие первичных функций от "обычных" функций, описанных на рефале, состоит в том, что обращения к первичным функциям всегда полностью вычисляются за один шаг рефал-машины (даже если при этом совершается длинная и сложная последовательность действий), а также в том, что при выполнении этих обращений может возникать побочный эффект, т.е. взаимодействие со средой, внешней по отношению к рефал-машине. В частности, с помощью первичных функций реализуются операции ввода/вывода.

I. ФУНКЦИЯ CARD

Первичная функция CARD дает возможность читать записи из входного файла с DD-именем SYSIN.

Файл SYSIN должен быть описан с помощью предложения FILEDEF (для CMS или ПДО), либо с помощью DD-предложения (для OS/360 или ОС ЕС).

Записи в файле SYSIN должны быть фиксированной или переменной длины, длиной не более 80 литер. Записи длиной менее 80 литер дополняются до 80 литер пробелами справа.

Обращение к CARD имеет следующий вид:

```
<CARD>
```

и выполняется следующим образом.

Если файл SYSIN еще не открыт - он открывается.

Если файл SYSIN пуст или уже прочитан до конца, работа функции CARD заканчивается, а результатом замены является пустое выражение.

Если файл еще не дочитан до конца, CARD читает очередную запись и выдает в качестве результата замены первые 80 литер этой записи.

Например, следующая функция CARDS читает всю входную информацию из SYSIN до конца и оставляет ее в поле зрения.

```
CARDS    = <CARDSI <CARD>>
CARDSI   VX = (VX) <CARDSI <CARD>>
         =
```

2. ФУНКЦИИ ПЕЧАТИ

Для печати результатов счета, можно использовать первичные функции PROUT и PRINT . Обращение к ним имеет следующий вид:

```
<PROUTM [E]>
<PROUTC [E]>
<PROUT  [E]>
<PRINTM [E]>
<PRINTC [E]>
<PRINT  [E]>
```

где [E] – произвольное об'ектное выражение.

В результате вычисления этих термов, в выходной файл SYSPRINT, начиная с новой строки, выводится выражение [E]. Если выражение не помещается в одной печатной строке, оно продолжается на следующих строках. Печать пустого [E] вызывает пропуск одной печатной строки.

Длина строки устанавливается при открытии файла SYSPRINT с помощью опции LINESIZE и должна быть не менее 120 литер. Если файл открывается неявно, автоматически устанавливается длина строки равная 120 литерам.

При использовании функций PRINTM и PROUTM выражение выводится в том же виде, в котором оно записывается в исходных рефал-программах. Единственное различие заключается в том, что при переходе на следующую запись не

ставится признак продолжения "+".

Функции PRINTC и PROUTC работают так же как и PRINTM и PROUTM, за исключением того, что они не обрамляют тела составных символов знаками "/".

При использовании функций PRINT и PROUT выражение [E] выводится следующим образом. Символы-литеры (об'ектные знаки) выводятся в виде соответствующих литер. Структурные скобки "(" и ")" выводятся в виде литер "(" и ")". Составные символы печатаются в том же виде, как они изображаются в рефал-программах, за исключением того, что в качестве ограничителей используется не "/", а апостроф "'".

Результатом вызова функции PROUTM, PROUTC или PROUT является пустое выражение. Результатом вызова функции PRINTM, PRINTC или PRINT является выражение [E].

3. ФУНКЦИИ АРИФМЕТИКИ

В этом разделе описаны функции для работы с целыми числами произвольной разрядности.

Для того, чтобы работать с целыми числами неограниченной разрядности, прежде всего необходимо представить их в виде об'ектных выражений. Все функции, описанные в этом разделе, используют следующее представление чисел.

Целое число изображается в виде последовательности символов-чисел (макроцифр), перед которой может стоять символ-литера '+' или '-'. Если знак числа отсутствует, то подразумевается знак '+'.

Примеры целых чисел:

```
'+' /17/ /242/ /1503/ /5/
    /17/ /242/ /1503/ /5/
'-' /2/
    /0/
'-' /3/ /2035/
```

Такое представление числа является записью в позиционной системе счисления по основанию 2**24=16777216 (2 в степени 24). Таким образом, три символа

```
'-' /3/ /2035/
```

изображают число

```
-(3 * 2**24 + 2035) = 50333683
```

Особый вопрос - представление нуля. Нуль может быть представлен либо выражением, содержащим только нулевые макроцифры, либо пустым выражением.

Все функции, описанные в данном разделе, допускают использование пустого выражения в качестве аргумента и воспринимают его как нуль. При этом, некоторые функции выдают нулевой результат работы в виде макроцифры /0/, а некоторые - в виде пустого выражения.

Функции ADD , MUL , SUB , DR , DIV .

Функции ADD , SUB , MUL и DR служат для выполнения сложения, вычитания, умножения, деления с остатком и деления нацело соответственно. Обращение к ним имеет следующий вид:

```
< [F] ( [N1] ) [N2] >
```

где [F] - имя функции, а [N1] и [N2] - целые числа (может

быть - пустые).

Результатом вызова функций ADD , SUB , MUL является целое число. Если результат положителен - знак '+' не ставится. Нулевой результат выдается в виде одной макроцифры /0/ . Например:

```
<ADD (/1/) /2/>      →    /3/
<ADD (  ) /2/>       →    /2/
<SUB (/1/) /2/>      →  '-'/1/
<SUB (/2/) /2/>      →    /0/
<MUL (/2/) '-'/2/>   →  '-'/4/
<MUL (/2/)>          →    /0/
```

Используя эти функции можно, например, описать функцию FAC , вычисляющую факториал неотрицательного целого числа.

```
FAC     EA = <FACI (EA) /1/>
FACI    (/0/) EB = EB
        (EA) EB = <FACI (<SUB (EA)/1/>) <MUL (EA)EB>>
```

Функция DR выдает результат в виде

[Q] ([R])

где [Q] - частное от деления [N1] на [N2] , а [R] - остаток.

И частное, и остаток выдаются без незначащих нулей и без знака '+'. Нулевые [Q] и [R] выдаются в виде макроцифры /0/.

Попытка делить на ноль приводит к авосту "отождествление невозможно".

Знаки чисел учитываются следующим образом: сначала производим деление, не обращая внимания на знаки, а затем частному и остатку приписываем такие знаки, чтобы

выполнялось соотношение:

$$[N1] = [Q] * [N2] + [R]$$

Т.е. частное положительно, если знаки делимого и делителя совпадают, и отрицательно в противном случае, а не равный нулю остаток всегда имеет знак делимого.

Например:

```
<DR (/5/) /3/>          → /1/ (/2/)
<DR (/5/) '-'/3/>       → '-'/1/ (/2/)
<DR ('-'/5/) /3/>       → '-'/1/ ('-'/2/)
<DR ('-'/5/) '-'/3/>    → /1/ ('-'/2/)
```

Функция DIV выдает результат в виде

$$[Q]$$

где [Q] - частное от деления [N1] на [N2] , точно такое же, какое получается при обращении к функции DR . Например:

```
<DIV (/5/) /3/>         → /1/
<DIV (/5/) '-'/3/>      → '-'/1/
<DIV ('-'/5/) /3/>      → '-'/1/
<DIV ('-'/5/) '-'/3/>   → /1/
```

Ф у н к ц и и ADDN , SUBN , MULN , DRN , DIVN .

В некоторых случаях удобно представлять нуль не макроцифрой /0/ , а пустым выражением. Для работы с таким представлением чисел предоставляются функции ADDN , SUBN , MULN , DRN . Они отличаются от ADD , SUB , MUL и DR только тем, что всегда выдают нулевой результат в виде пустого выражения. Например:

```
<MULN () /3/>           → [пусто]
```

```
<SUBN (/0/)>          → [пусто]
<DRN  (/10/) /5/>     → /22/ ()
<DRN  () /3/>         → ()
<DIVN () /3/>         → [пусто]
```

Функция GCD.

Функция GCD предназначена для вычисления наибольшего общеего делителя двух целых чисел. Формат обращения:

```
<GCD ( [N1] ) [N2] >
```

где [N1] и [N2] – целые числа (которые могут быть пустыми). Результат замены имеет вид:

```
[N]
```

где [N] – положительное целое число, которое является наибольшим общим делителем чисел [N1] и [N2]. Например:

```
<GCD (/6/) /15/>      → /3/
<GCD ('-'/6/) /15/>   → /3/
<GCD (/15/) /1/>      → /1/
<GCD (/15/) /0/>      → /15/
<GCD (/15/)>          → /15/
```

Наибольший общий делитель не может быть нулем, поэтому не возникло необходимости иметь еще и функцию GCDN.

Если и [N1], и [N2] равны нулю, возникает авост "отождествление невозможно".

Используя функцию DR, можно было бы описать GCD на рефале следующим образом:

```
GCD     (E1) EA (/0/) = E1
        (E1) EA (E2)  = <GCD (E2) <DR (E1) E2>>
```

Ф у н к ц и я NREL .

Функция NREL предназначена для сравнения целых чисел. Формат обращения:

<NREL ([N1]) [N2] >

где [N1] и [N2] - целые числа или пустые выражения. Результат замены имеет вид:

[Z] ([N1]) [N2]

где [Z] - символ-литера. При этом

Если [N1] > [N2] , то [Z] = '>' .

Если [N1] = [N2] , то [Z] = '=' .

Если [N1] < [N2] , то [Z] = '<' .

Числа [N1] и [N2] сравниваются как целые с учетом знака. При этом пустое выражение воспринимается как нуль. Например:

```
<NREL (/5/) /3/>          ==> '>' (/5/) /3/
<NREL ('-'/5/) '-'/3/>    ==> '<' ('-'/5/) '-'/3/
<NREL () /0//0/>          ==> '=' () /0//0/
```

Функции NUMB , SYMB , CVB и CVD применяются для перевода чисел из десятичной системы в систему по основанию 2**24 и обратно.

Ф у н к ц и я NUMB .

Для перевода чисел, не превосходящих по абсолютной величине 2**31-1 = 2147483647, из десятичной системы в систему по основанию 2**24 используется первичная функция NUMB . Обращение к ней имеет следующий вид:

```
<NUMB [D]>
```

где [D] – цепочка символов-литер, являющаяся десятичной записью целого числа со знаком или без него, или пустое выражение. Результат замены – то же число в системе по основанию 2**24. Пустое [D] воспринимается как нуль. Нулевой результат изображается как /0/ . Например:

```
<NUMB '1000'>            ➔    /1000/
<NUMB '-16777216'>       ➔    '-' /1/ /0/
<NUMB>                   ➔    /0/
<NUMB '1000000000'>      ➔    /59/ /10144256/
<NUMB '+025'>            ➔    /25/
```

Ф у н к ц и я SYMB .

Для перевода чисел, не превышающих по абсолютной величине 2**31-1 = 2147483647, из системы по основанию 2**24 в десятичную систему используется функция SYMB . Обращение к ней имеет следующий вид:

```
<SYMB [N]>
```

где [N] – целое число в системе по основанию 2**24 (которое может быть пустым). Результат замены – цепочка символов-литер, являющаяся записью числа в десятичной системе счисления. Нулевой результат изображается как '0' . Например:

```
<SYMB /1000/>            ➔    '1000'
<SYMB '-' /1/ /0/>       ➔    '-16777216'
<SYMB>                   ➔    '0'
<SYMB /0/>               ➔    '0'
<SYMB /59/ /10144256/>   ➔    '1000000000'
<SYMB '-' /0/ /25/>      ➔    '-25'
```

Функция CVB .

Для перевода целых чисел произвольной величины из десятичной системы в систему по основанию 2**24 используется функция CVB . Обращение к ней имеет следующий вид:

<CVB [D]>

где [D] - цепочка символов-литер, являющаяся десятичной записью целого числа со знаком или без него, или пустое выражение. Результат замены - то же число в системе по основанию 2**24. Пустое [D] воспринимается как нуль. Нулевой результат изображается как /0/ .

Функция CVD .

Для перевода целых чисел произвольной величины из системы по основанию 2**24 в десятичную систему используется функция CVD . Обращение к ней имеет следующий вид:

<CVD [N]>

где [N] - целое число в системе по основанию 2**24 (которое может быть пустым). Результат замены - цепочка символов-литер, являющаяся записью числа в десятичной системе счисления. Нулевой результат изображается как '0' .

Функции CVB и CVD описаны на рефале следующим образом:

```
        START
        ENTRY   CVB,CVD
        EXTRN   NUMB,SYMB
        EXTRN   ADD,MUL,DRN
CVB     '-' V(D)X = '-' <CVB0 VX>
        '+' V(D)X =     <CVB0 VX>
            E(D)X =     <CVB0 EX>
```

```
CVB0    VX S1 S2 S3 S4 S5 S6 S7 S8 S9 = +
          <ADD (<MUL (<CVB0 VX>) +
          /59/ /10144256/>)              +
          <NUMB S1 S2 S3 S4 S5 S6 S7 S8 S9>>
        EX = <NUMB EX>
CVD     '-' V(N)X = '-' <CVD0 VX>
        '+' V(N)X =     <CVD0 VX>
            E(N)X =     <CVD0 EX>
CVD0    EX = <CVD1 <DRN (EX) /59/ /10144256/>>
CVD1    (EY) = <SYMB EY>
        EX (EY) = +
          <CVD0 EX> +
          <CVD2 <SYMB EY>>
CVD2    S1 S2 S3 S4 S5 S6 S7 S8 S9 = +
          S1 S2 S3 S4 S5 S6 S7 S8 S9
        EX = <CVD2 '0' EX>
        END
```

Обратите внимание на то, что /59/ /10144256/ является представлением целого числа 1000000000.

4. ФУНКЦИИ ЛЕКСИЧЕСКОГО АНАЛИЗА

Функция TYPE.

Функция TYPE предназначена для распознавания типа терма, с которого начинается выражение. Эта функция не дает никаких новых возможностей по сравнению с теми, которые дает использование спецификаторов. Она включена в библиотеку только для того, чтобы обеспечить совместимость с другими реализациями рефала, и имеет следующее описание на рефале-2.

```
        START
        ENTRY   TYPE
TYPE    S(F)X E1 = 'F' SX E1
        S(N)X E1 = 'N' SX E1
        S(R)X E1 = 'R' SX E1
        S(L)X E1 = 'L' SX E1
        S(D)X E1 = 'D' SX E1
        S(O)X E1 = 'O' SX E1
        (EX) E1 = 'B' (EX) E1
        = '*'
        END
```

Ф у н к ц и я FIRST .

Эта функция предназначена для отщепления от начала выражения части, имеющей указанную длину. Обращение к ней имеет вид.

<FIRST [N] [E]>

где [N] - макроцифра, а [E] - произвольное об'ектное выражение.

Если [E] представляет собой последовательность из менее, чем [N] термов, результатом замены является выражение

'*' [E]

Если же длина [E] не меньше, чем [N] термов, результатом замены является

([E1]) [E2]

где [E1] и [E2] - такие выражения, что [E]=[E1] [E2] и при этом [E1] является последовательностью из ровно [N] термов.

Например:

```
<FIRST/2/ 'A'('B')'C' > ⟶ ('A'('B'))'C'
<FIRST /5/ 'A'('B')'C'> ⟶  '*A'('B')'C'
```

Ф у н к ц и я LAST .

Эта функция предназначена для отщепления от конца выражения части, имеющей указанную длину. Обращение к ней имеет следующий вид:

```
<LAST [N] [E] >
```

где [N] - макроцифра, а [E] - произвольное об'ектное выражение.

Если [E] представляет собой последовательность из менее, чем [N] термов, результатом замены является

```
[E] '*'
```

Если же длина [E] составляет не менее, чем [N] термов, результатом замены является

```
[E1] ( [E2] )
```

где [E1] и [E2] - такие выражения, что [E]=[E1] [E2] и при этом [E2] является последовательностью из ровно [N] термов. Например:

```
<LAST /2/ 'A'('B')'C'> ⟶  'A'(('B')'C')
<LAST /5/ 'A'('B')'C'> ⟶  'A'('B')'C*'
```

П р и м е р. Опишем функцию CARD72 , которая читает очередную запись из файла SYSIN и берет из нее первые 72 литеры, а остальные - отбрасывает.

```
        EXTRN CARD
CARD72  = <CARD72X <LAST /8/ <CARD>>>
CARD72X E1 (E2) = E1
        '*' =
```

Напоминаем, что когда CARD достигает конца файла она выдает пустое выражение в качестве результата замены. Именно этот случай предусмотрен во втором предложении функции CARD72X .

Функции LENGW и LENGR .

Эти функции предназначены для вычисления длины аргумента. Обращение к ним имеет следующий вид:

```
<LENGW [E]>
<LENGR [E]>
```

где [E] - произвольное об'ектное выражение.

Результатом замены является выражение

```
[N] [E]
```

где [N] - макроцифра, которая равна длине выражения [E] . При этом функция LENGW выдает длину выражения, измеренную в термах, т.е. количество термов нулевого уровня, составляющих [E] , а функция LENGR - выдает "реальную" длину выражения, т.е. количество символов и скобок, составляющих [E] . Например:

```
<LENGW 'A' () ('A')>  → /3/ 'A' () ('A')
<LENGR 'A' () ('A')>  → /6/ 'A' () ('A')
<LENGW>               → /0/
```

Ф у н к ц и я MULTE .

Эта функция копирует (размножает) выражение в заданном количестве экземпляров. Обращение к ней имеет вид.

<MULTE [N] [E]>

где [N] - макроцифра, а [E] - произвольное об'ектное выражение.

Результатом замены является

[E] [E] ... [E]

где выражение [E] повторяется [N] раз. В частности, если [N] равно нулю, результатом замены является пустое выражение. Например:

<MULTE /5/ 'A'> → 'AAAAA'
<MULTE /2/ 'A'('B')> → 'A'('B')'A'('B')
<MULTE /0/> → [пусто]

Во всякой реализации рефала на множестве символов существует отношение порядка, определяемое кодировкой, принятой в этой реализации. Этим отношением можно воспользоваться с помощью функции CPSYMB .

Ф у н к ц и я CPSYMB .

Функция CPSYMB предназначена для сравнения символов. Формат обращения:

<CPSYMB [A] [B]>

где [A] и [B] - символы. Результат замены имеет вид:

[Z]

где Z – символ-литера. При этом

если [A] > [B] , то [Z] = '>' ;

если [A] = [B] , то [Z] = '=' ;

если [A] < [B] , то [Z] = '<' .

Символы сравниваются по их представлению, использованному в данной реализации рефала. Это значит, что классы символов упорядочены следующим образом (в порядке возрастания):

- символы-литеры,
- символы-метки,
- символы-числа,
- символы-ссылки.

Сами символы-литеры упорядочены по коду EBCDIC ; символы-метки – по адресам расположения функций в памяти ЭВМ (внутри одного модуля – в порядке описания функций, между модулями – в зависимости от последовательности загрузки в память ЭВМ); символы-числа – по значению чисел; символы-ссылки – по адресам голов ящиков (которые видны на печати).

Например:

```
<CPSYMB 'A' 'Z'>        → '<'
<CPSYMB /2/ /1/>        → '>'
<CPSYMB 'A' /XXX/>      → '<'
<CPSYMB /XXX/ /0/>      → '<'
```

5. ФУНКЦИИ PI И MI

Первичные функции PI и MI производят прибавление и вычитание единицы из макроцифры. Обращение к ним имеет следующий вид:

```
<PI [N]>
<MI [N]>
```

где [N] - макроцифра.

Результатом замены для PI является макроцифра, на единицу большая, чем [N] . Результатом замены для MI является макроцифра, на единицу меньшая, чем [N] . Например:

```
<PI /1/>     → /2/
<PI /379/>   → /380/
<MI /10/>    → /9/
<MI /7382/>  → /7381/
<MI /1/>     → /0/
```

Попытка прибавить единицу к макроцифре /16777215/ или вычесть единицу из макроцифры /0/ приводит к авосту "отождествление невозможно".

6. ФУНКЦИИ ДЛЯ РАБОТЫ С СИМВОЛЬНЫМИ ФАЙЛАМИ

В этом разделе описаны функции LIBGET , OPNGET , CLSGET , LIBPUT , OPNPUT , CLSPUT , предназначенные для работы с последовательными файлами. Читаться могут файлы с записями фиксированной и переменной длины, содержащими не более 80 литер. Записи длиной менее 80 литер дополняются справа пробелами до 80 литер. Записывать можно только в файлы с записями фиксированной длины по 80 литер.

Каждый файл, с которым будет работать рефал-программа, должен быть описан перед ее запуском с помощью предложения FILEDEF (для CMS или ПДО), либо с помощью DD-предложения (для OS/360 или ОС ЕС). При этом должны быть указаны параметры DCB: RECFM и LRECL. Например:

```
FILEDEF XXX DISK AAA PLIOPT A (RECFM F LRECL 80

//XXX  DD   DSN=LIB.SYM(MEMBER1),DISP=SHR,
            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
            UNIT=2311,VOL=SER=AUTOL
```

Описанные в этом разделе функции позволяют работать не более чем с двумя файлами одновременно. При этом один из этих файлов доступен только на чтение (и именуется "файлом чтения"), а другой файл – доступен только на запись (и именуется "файлом записи").

Чтение файлов производится с помощью первичных функций LIBGET , OPNGET и CLSGET .

Функция OPNGET .

Прежде чем начинать чтение файла, его следует открыть с помощью функции OPNGET . Обращение к ней имеет следующий вид:

<OPNGET [DDNAME]>

где [DDNAME] – это цепочка символов-литер, которая представляет собой имя того предложения FILEDEF или DD, в котором описан открываемый файл.

Функция LIBGET .

После того как файл открыт, можно читать из него записи с помощью функции LIBGET . Обращение к ней имеет следующий вид:

<LIBGET>

В результате обращения функция LIBGET пытается прочитать очередную запись из файла. Если файл прочитан до конца, результатом замены является пустое выражение. Если же в файле еще остались непрочитанные записи, LIBGET читает очередную запись и выдает ее в качестве результата замены в виде цепочки из 80 символов-литер.

Если в момент обращения к LIBGET файл чтения не был открыт, LIBGET автоматически открывает файл, для которого [DDNAME]='SYSGET'.

Функция CLSGET .

После того, как работа с файлом чтения закончена, нужно закрыть его с помощью функции CLSGET . Обращение к ней имеет следующий вид

<CLSGET>

Результат замены - пустое выражение.

После того, как файл чтения закрыт, можно вновь обратиться к функции OPNGET и начать чтение другого файла. В частности, можно прочитать повторно тот же самый файл.

Запись в файлы производится с помощью функций LIBPUT , OPNPUT , CLSPUT .

Функция OPNPUT .

Прежде, чем что-либо писать в файл, его следует открыть с помощью первичной функции OPNPUT . Обращение к ней имеет следующий вид:

<OPNPUT [DDNAME]>

где [DDNAME] - цепочка символов-литер, которая представляет собой имя того предложения FILEDEF или DD, в котором описан открываемый файл.

Функция LIBPUT .

После того, как файл открыт, можно писать в него записи с помощью функции LIBPUT , обращение к которой имеет следующий вид:

<LIBPUT [E]>

где [E] - об'ектное выражение, которое не содержит составных символов.

В результате обращения к LIBPUT в выходной файл записывается вообще говоря несколько записей, которые формируются следующим образом. Сначала все структурные

скобки в выражении [E] преобразуются в символы-литеры '(' и ')'. Затем получившаяся цепочка символов-литер разбивается на куски по 80 литер. Если последний кусок содержит менее 80 литер, он дополняется до 80 литер пробелами справа. Если в результате этих действий получается N кусков, в выходной файл записывается N записей, каждая длиной в 80 литер. В частности, если аргумент [E] пуст, в файл добавляется нулевое число записей, т.е. состояние файла не изменяется.

Результатом замены для LIBPUT всегда является пустое выражение.

Если в момент обращения к LIBPUT файл записи еще не открыт, LIBPUT автоматически открывает на запись файл, для которого [DDNAME]='SYSPUT'.

Функция CLSPUT.

После того как работа с файлом записи закончена, следует закрыть его с помощью функции CLSPUT, обращение которой имеет следующий вид:

<CLSPUT>

Результатом замены для CLSPUT всегда является пустое выражение.

После того как файл записи закрыт, можно открыть его в качестве файла чтения с помощью функции OPNGET и прочитать его содержимое. Кроме того, можно открыть на запись другой файл и записывать в него информацию.

7. ФУНКЦИЯ APPLY

Во многих случаях требуется, чтобы при возникновении аварийной ситуации "отождествление невозможно" или "свободная память исчерпана" рефал-программа не прекращала свою работу, а анализировала аварийную ситуацию и предпринимала какие-то дальнейшие действия. Для таких случаев предусмотрена функция APPLY .

Ф у н к ц и я APPLY .

Обращение к этой функции из рефал-программы имеет следующий вид:

<APPLY [E] >

где [E] - произвольное выражение.

Выполнение этого вызова происходит следующим образом. Создается новое поле зрения, в которое помещается функциональный терм

< [E] >

После этого делается попытка вычислить этот функциональный терм.

Возможны три исхода этой попытки: нормальный останов (N), останов "отождествление невозможно" (R) и останов "свободная память исчерпана" (S).

В случае N результатом замены будет выражение

'N' [E1]

где [E1] - результат вычисления терма <[E]> .

В случае R результатом замены будет выражение

'R' [E1]

где [E1] - содержимое того функционального терма вида <[E1]>, попытка вычисления которого привела к авосту "отождествление невозможно".

В случае S результатом замены будет выражение

'S'

После того, как результат замены сформирован, дополнительное поле зрения, созданное в результате обращения к APPLY , уничтожается.

Можно обращаться к APPLY рекурсивно. В этом случае образуется стек из полей зрения.

8. ФУНКЦИИ ДЛЯ РАБОТЫ С СИМВОЛАМИ-МЕТКАМИ

В некоторых случаях возникает необходимость превратить цепочку литер в символ-метку или, наоборот, получить из символа-метки цепочку литер, составляющих его тело. Например, из цепочки литер 'ABCD-EF' построить символ /ABCD-EF/ или, например, символ /A-123/ превратить в цепочку 'A-123'.

При этом должно выполняться следующее требование: если мы будем превращать одну и ту же цепочку литер в символ-метку

несколько раз, то все созданные символы-метки должны быть равны друг другу, т.е. являться экземплярами одного и того же символа.

Для создания и расчленения символов-меток используются функции FTOCHAR и CHARTOF .

Ф у н к ц и я FTOCHAR .

Эта функция превращает символ-метку в цепочку литер, которая является телом символа-метки. Обращение к ней имеет вид:

<FTOCHAR [F]>

где [F] - символ-метка. Результатом замены является цепочка литер, являющаяся телом символа [F]. Например:

<FTOCHAR /A-123/> → 'A-123'
<FTOCHAR /ABCD/> → 'ABCD'

Ф у н к ц и я CHARTOF .

Эта функция превращает цепочку литер в символ-метку, имеющий тело, совпадающее с заданной цепочкой литер. Одна и та же цепочка литер превращается в один и тот же символ-метку. Обращение к функции имеет вид:

<CHARTOF [C]>

где [C] - цепочка литер. Результатом замены является символ-метка. Например:

<CHARTOF 'A-123'> → /A-123/
<CHARTOF 'ABCD'> → /ABCD/

Функция CHARTOF создает и поддерживает таблицу

символов-меток, порожденных ею из цепочек литер. Если цепочка литер встретилась в первый раз, то создается пустая функция, имя которой является соответствующим символом-меткой. Если такая же цепочка литер встречается еще раз, то новая функция не создается, а в качестве результата вырабатывается имя ранее созданной функции.

В некоторых случаях требуется, чтобы символ-метка, создаваемый из цепочки литер [C] заведомо совпадал с символом-меткой, определенным в рефал-программе и имеющим тело [C]. Это достигается с помощью функции FUNCTAB .

Функция FUNCTAB .

Эта функция регистрирует в таблице символов, создаваемой функцией CHARTOF , указанный символ-метку. Обращение к ней имеет следующий вид:

<FUNCTAB [F]>

где [F] - символ-метка. Результатом замены является [пусто]. Символ [F] регистрируется в таблице символов. После этого, всякий раз, когда функция CHARTOF будет получать цепочку литер, совпадающую с телом символа [F], результатом функции CHARTOF будет символ [F].

Функции CHARTOF и FUNCTAB нуждаются в памяти для построения таблицы символов, поэтому может потребоваться увеличить количество памяти, заказываемой посредством опции ISASIZE при запуске рефал-программы.

9. ФУНКЦИИ ДЛЯ РАБОТЫ С ЛЕКСИКОГРАФИЧЕСКИ УПОРЯДОЧЕННЫМИ МНОЖЕСТВАМИ ВЫРАЖЕНИЙ

В этом разделе описываются функции, использующие понятия статического и динамического ящиков для решения следующей задачи. Во многих приложениях Рефала (например, в аналитических вычислениях на ЭВМ) требуется упорядочивать выражения по некоторому отношению порядка и строить составные данные из частей с учетом этого порядка. (Пример: выполнение операций над полиномами в "нормализованной" форме.) В таких алгоритмах сравнение выражений является одной из частых операций.

Программируя операции над упорядоченными выражениями, обычно заводят вспомогательную функцию двух аргументов, имеющую примерно такой формат вызова:

$$\langle \text{COMPARE } ([E1])\ ([E2]) \rangle \rightarrow [Z]$$

где [Z] - это символ-литера '<', '=' или '>'.

Такая функция сравнивает выражения [E1] и [E2] по зафиксированному в ней отношению порядка.

Вычисление значения функции сравнения COMPARE часто требует достаточно много шагов рефал-машины (особенно, если [E1] и [E2] "длинные" и "почти совпадают": ведь даже при простейшем лексикографическом порядке нужно последовательно просмотреть совпадающие части).

С другой стороны, статистика использования функций сравнения, как правило, такова, что значения ее аргументов порождаются гораздо реже, чем они сравниваются между собой. (Например, самый быстрый алгоритм сортировки N

"неизменных" элементов последовательности сравнивает между собой N*LOG(N) раз.)

В связи с этим возникает вопрос: а нельзя ли работу по анализу выражений для выяснения их места в упорядоченном множестве перенести из стадии попарных сравнений на период их порождения, чтобы собственно сравнения вычислялись "быстро"?

Ответ: можно; только надо где-то хранить результаты этого предварительного анализа.

В Рефале-2 понятие "места хранения" реализуется статическими и динамическими ящиками. Понятие ящика ("об'екта" - если использовать терминологию об'ектно-ориетированного подхода в программировании) обладает одним важным свойством: в ящик можно поместить любую информацию, а затем в качестве представителей этой информации в данных и аргументах функций использовать с и м в о л ы - с с ы л к и , которые сами по себе никак не отражают вида этой информации; доступ к содержимому ящиков можно ограничить фиксированным набором функций, в которых будет с к р ы т о представление данных и любой дополнительной информации, хранимой в ящиках. Здесь важно наличие четкого разделения на момент создания ящика с занесением в него значения и моменты использования содержимого ящика в последующих вычислениях. В момент формирования ящика можно один раз вычислить и запасти в ящике любую дополнительную информацию: "снаружи" этот факт никак не будет виден.

Будем называть т а б у л я ц и е й такой стиль использования ящиков, когда помещаемая в них информация в дальнейшем не уничтожается, хранится все время "жизни" ящика и используется в фиксированном наборе функций от содержимого ящика. Про эти функции и будем говорить, что они "табулированы" с помощью данных ящиков.

В чем может заключаться "табуляция отношения порядка"? Иначе: по какой информации его можно вычислить "быстро"? Для этого достаточно вместе со значением элемента упорядоченного множества хранить в ящике некоторый номер, который отражает его место в отношении порядка среди тех элементов, для которых уже порождены ящики, и затем вместо самих значений сравнивать эти небольшие целые числа. Эти номера, конечно, должны быть скрыты от пользовтеля: более того, они могут меняться, т.к. время от времени возникает необходимость в перенумерации.

В существующей версии поддерживается только одно - л е к с и к о г р а ф и ч е с к о е отношение порядка с некоторыми дополнительными "рычагами управления" порядком. (В будущем предполагается ввести средства описания отношения порядка.) Реализованный лексикографический порядок выражений использует следующий порядок элементов выражения (в порядке возрастания):

- правый конец выражения;
- ")" - правая структурная скобка;
- "(" - левая структурная скобка;
- символы, кроме ссылок (сами они упорядочены с помощью функции CPSYMB);
- символы ссылки на ящики, представляющие элементы упорядоченных множеств (они сами упорядочены по присущему им отношению порядка).

В выражениях, подлежащих упорядочению, нельзя использовать символы-ссылки на ящики, созданные другими функциями, кроме описанных в этом разделе.

Для структуризации совокупности упорядоченных ящиков она

разбивается на конечное число с е м е й с т в . В качестве и м е н семейств используются имена с т а т и ч е с к и х я щ и к о в (которые для других целей использовать нельзя, т.к. в них хранится некоторая информация, относящаяся к данному семейству). Таким образом, чтобы определить упорядоченные семейства, нужно в программе написать одно или несколько предложений вида:

SWAP [F]

где [F] - имя семейства (имя статического ящика).

Элементы различных семейств сравниваются по именам семейств (с помощью функции CPSYMB): больше то, которое об'явлено ниже по тексту программы.

Имена описанных ниже функций начинаются с букв RO - от слов REFERENCE (ссылка) и ORDER (порядок), что означает, что речь идет о выполнении операций над элементами упорядоченных множеств, представленных символами-ссылками на ящики, их содержащие.

Ф у н к ц и я ROTAB - "табулировать элемент упорядоченного множества":

<ROTAB [F] [E]> → [R]

где [F] - символ-метка - имя семейства, [E] - значение элемента семейства, [R] - символ-ссылка на ящик, представляющий элемент семейства. Выражение [E] не должно содержать символов-ссылок кроме тех, которые созданы ранее функцией ROTAB .

Эта функция заносит значение [E] в семейство [F] и выдает представитель этого элемента - символ-ссылку [R]. При повторном обращении к ROTAB с такими же значениями [F]

и [E] выдается ссылка на тот же самый ящик (кроме случаев, когда тот ящик был уничтожен сборкой мусора из-за отсутствия ссылок на него).

Функция ROREL сравнивает два символа-ссылки на элементы упорядоченных семейств, сформированные функцией ROTAB :

<ROREL [R] [Q]> → [Z]

где [R] и [Q] - символы-ссылки, [Z] - символ-литера '<', '=' или '>' ('=' выдается тогда и только тогда, когда [R]=[Q]).

Значение функции ROREL вычисляется за время, сравнимое со средним шагом рефал-машины.

Функции ROFAM и ROVAL - выдают имя семейства и значение элемента, соответственно:

<ROFAM [R]> → [F]
<ROVAL [R]> → [E]

где [R] - символ-ссылка, сформированный функцией ROTAB. результате обращения к ней в виде:

<ROTAB [F] [E]>

Именно эти [F] и [E] выдаются функциями ROFAM и ROVAL .

Кроме использования встроенного лексикографического отношения порядка, в каждом семействе можно об'явить минимальный и максимальный элементы.

Функции ROMIN и ROMAX имеют такой формат:

<ROMIN [F] [E]> → [пусто]
<ROMAX [F] [E]> → [пусто]

где [F] - имя семейства, [E] - значение элемента семейства.

При первом обращении функции ROMIN (или ROMAX) к семейству [F] в нем запоминается, что отныне [E] будет значением минимального (соответственно: максимального) элементов данного семейства. В дальнейшем при вызове <ROTAB [F] [E]> будет выдан символ-ссылка на ящик, представляющий этот элемент.

Если к функции ROMIN (или ROMAX) обратиться несколько раз, то формируется начальный (соответственно: конечный) отрезок элементов семейства: для ROMIN - в порядке возрастания, для ROMAX - в порядке убывания.

При использовании ROMIN и ROMAX возникает конфликт, если обращение к одной из них происходит после того, как уже был вызов ROTAB, ROMIN или ROMAX с такими же [F] и [E]. Этот конфликт разрешается следующим образом.

Если элемент со значением [E] уже есть в списке минимальных или максимальных элементов данного семейства, тот новое обращение к ROMIN или ROMAX игнорируется.

Если с одинаковыми аргументами последовательно вызывались ROTAB, ROMIN (или ROMAX), а затем снова ROTAB, то два обращени к ROTAB выдадут не совпадающие символы-ссылки на различные элементы семейства; при этом первый будет сформирован в соответствии с лексикографическим отношением порядка, а второй - по указанию функции ROMIN (или ROMAX).

Функция RORESET очищает указанные семейства:

< RORESET [F1] ... [FN] > → [пусто]

где [F1] ... [FN] – последовательность имен семейств.

В результате обращения к функции RORESET у указанных семейств забываются списки минимальных и максимальных элементов и все ящики, представляющие элементы семейства, переводятся в специальное состояние, которое приведет к авосту при использовании их в аргументах функций, описанных в этом разделе. Ящики – элементы семейств, на которые нет ссылок, будут в дальнейшем удалены из памяти ЭВМ очередной сборкой мусора.

БЛАГОДАРНОСТЬ

Авторы выражают благодарность Е.В.Травкиной, выполнившей трудоемкую работу по реализации функций арифметики на языке ассемблера.

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.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ ФУНКЦИЙ

<PRINT [E]>	→	[E]	5
<PRINTC [E]>	→	[E]	5
<PRINTM [E]>	→	[E]	5
<PROUT [E]>	→	[пусто]	5
<PROUTC [E]>	→	[пусто]	5
<PROUTM [E]>	→	[пусто]	5
<PI [N]>	→	[N+1]	20
<ROFAM [R]>	→	[F]	29
<ROMIN [F] [E]>	→	[пусто]	29
<ROMAX [F] [E]>	→	[пусто]	29
<ROTAB [F] [E]>	→	[R]	29
<ROVAL [R]>	→	[E]	29
<ROREL [R1] [R2]>	→	[Z]	29
<RORESET [F1] ... [F2]>	→	[пусто]	29
<SUB ([N1]) [N2]>	→	[N]	7
<SUBN ([N1]) [N2]>	→	[N]	9
<SYMB [N]>	→	[D]	12
<TYPE [E]>	→	[Z] [E]	14

Климов Андрей Валентинович, Романенко Сергей Анатольевич " Система программирования Рефал – 2 для ЕС ЭВМ.Описание библиотеки функций."

Редактор В.С. Штаркман. Корректор А.О. Лацис.

Подписано к печати 08.12.86г. № Т- 21902. Заказ № 473.
Формат бумаги 60Х90 1/16. Тираж 430 экз.
Объём 1,7 уч.-изд.л. Цена 14 коп.

055 (02)2

Отпечатано на ротапринтах в Институте прикладной математики АН СССР

Москва, Миусская пл. 4.

Ссылки на издание рекомендуется делать по следующей форме: и.о., фамилия, название, препринт Ин. прикл. матем. им. М.В. Келдыша АН СССР, год, №.

Распространение: препринты института продаются в магазинах Академкниги г. Москвы, а также распространяются через Библиотеку АН СССР в порядке обмена.

Адрес: СССР, 125047, Москва-47, Миусская пл. 4, Институт прикладной математики им. М.В. Келдыша АН СССР, ОНТИ.

The references should be typed by the following form: initials, name, title, preprint, Inst.Appl.Mathem., the USSR Academy of Sciences, year, N(number).

Distribution. The preprints of the Keldysh Institute of Applied Mathematics, the USSR Academy of Sciences are sold in the bookstores "Academkniga", Moscow and are distributed by the USSR Academy of Sciences Library as an exchange.

Adress: USSR, I25047, Moscow A-47, Miusskaya Sq.4, the Keldysh Institute of Applied Mathematics, Ac.of Sc., the USSR, Information Bureau.

Цена 14 коп.