

УДК 681.3.06:51

СХЕМНО-СПИСОЧНАЯ СИМВОЛИКА В ЯЗЫКЕ
ДЕЛЬТА-ЭПСИЛОН

Л.Т. Петрова

Некоторые идеи крупноблочной схемной символики [1] - [4] применяются здесь для построения расширения одного из современных языков программирования ЭПСИЛОН [5]. Расширенный язык называется ДЕЛЬТА-ЭПСИЛОН. В языке ЭПСИЛОН, в частности, введены списки-объекты, подобные тем, которые рассматривались ранее в схемно-списочных вычислениях [3], [4]. Это обстоятельство облегчает реализацию в этом языке и некоторых других подходов схемно-списочных вычислений. Простота конструкций, средства для частичного планирования машинной памяти делают язык ЭПСИЛОН удобным инструментом для выполнения такого расширения: транслятор с языка ДЕЛЬТА-ЭПСИЛОН естественно записать на языке ЭПСИЛОН.

Итак, синтаксис языка ДЕЛЬТА-ЭПСИЛОН ($\Delta\epsilon$) строится на основе синтаксиса языка ЭПСИЛОН (ϵ). Формально в синтаксисе языка ЭПСИЛОН вводятся следующие дополнения:

- 1⁰ $\langle \text{описатель } \Delta\epsilon \rangle ::= \langle \text{описатель } \epsilon \rangle \mid \langle \text{схема} \mid \text{оператор} \rangle$
- 2⁰ $\langle \text{объект } \Delta\epsilon \rangle ::= \langle \text{объект } \epsilon \rangle \mid \langle \text{укрупненный объект} \rangle$
- 3⁰ $\langle \text{операция } \Delta\epsilon \rangle ::= \langle \text{операция } \epsilon \rangle \mid \langle \text{укрупненная операция} \rangle$
- 4⁰ $\langle \text{выражение } \Delta\epsilon \rangle ::= \langle \text{выражение } \epsilon \rangle \mid \langle \text{укрупненное выражение} \rangle$
- 5⁰ вводится новая конструкция $\langle \text{схема} \rangle$ и связывание с этим конструкции $\langle \text{список-схема} \rangle$ и $\langle \text{процедура-схема} \rangle$.
- 6⁰ выделяется $\langle \text{укрупненный оператор процедуры} \rangle$.

Отметим, что дополнения, перечисленные в пунктах 1⁰ - 4⁰ и 6⁰ являются расширением понятий и конструкций, уже имевшихся

в исходном языке, а в пункте 5⁰ указаны новые конструкции. Естественно, что расширение исходных понятий приводит к расширению семантического толкования некоторых операторов исходного языка, которые синтаксически могут даже и не измениться. В этой связи выделяется укрупненный оператор процедуры (7), имеющий специфические семантические особенности по сравнению с обычным оператором процедуры. Естественно также, что новый язык приобретает некоторые новые качества по сравнению с исходным. Так, если язык \mathcal{E} ориентирован на запись алгоритмов в алгольном стиле, т.е. на уровне преобразования атомарных единиц информации, то в языке $\Delta\mathcal{E}$ в записи алгоритмов могут выступать как единое целое более сложные образования (списки, процедуры, схемы). Можно указать и другие особенности нового языка. В частности, введение списков-схем дает возможность конструировать более сложные правила формирования очередной порции информации, чем обычная выборка элемента массива по его номеру или признаку. Наконец, здесь рассматриваются операции над процедурами, с помощью которых исходные процедуры некоторого класса согласуются и объединяются в одну общую более сложную процедуру того же класса (процедуры-схемы).

Перейдем к изложению синтаксиса и семантики новых синтаксических единиц языка.

§ 1. Расширение исходных понятий

1. Укрупненные объекты

1а. Синтаксис

$\langle \text{укрупненный объект} \rangle ::= \langle \text{список} \rangle \mid \langle \text{процедура} \rangle$

1б. Семантика

В отличие от языка \mathcal{E} , в языке $\Delta\mathcal{E}$ список и процедура являются объектами, и, следовательно, для них должны быть определены значения. Значением списка является совокупность значений его элементов. Значением процедуры является ее программный образ. Поскольку список и процедура объявлены теперь объектами, они могут фигурировать в выражениях (3.1 и 3.2).

2. Укрупненные операции

2а. Синтаксис

$\langle \text{укрупненная операция} \rangle ::= \langle \text{операция над списками} \rangle |$
 $\langle \text{операция над процедурами} \rangle$
 $\langle \text{операция над списками} \rangle ::= \otimes | \cup | \parallel$
 $\langle \text{операция над процедурами} \rangle ::= \text{композиция} | \text{объединение} |$
 $\text{соединение} | \text{разветвление} | \text{повторение}$

2с. Семантика

Операция \otimes называется прямым произведением, операция \cup - объединением, а \parallel - соединением списков. Смысл введенных операций над списками и над процедурами выясняется в 3. Формально они служат для образования списочных выражений (3.1) и процедурных выражений (3.2).

3. Укрупненные выражения

3а. Синтаксис.

$\langle \text{укрупненное выражение} \rangle ::= \langle \text{списочное выражение} \rangle |$
 $\langle \text{процедурное выражение} \rangle$

3.1. Списочные выражения

3.1а. Синтаксис

$\langle \text{списочное выражение} \rangle ::= \langle \text{имя списка} \rangle \langle \text{операция над списками} \rangle \langle \text{имя списка} \rangle$

3.1.в. Примеры

$A \otimes B$

$A \cup B$

$A \parallel B$

3.1.с. Семантика.

Для того, чтобы пояснить смысл списочных операций и выражений, обратимся к истолкованию оператора присваивания, в котором фигурирует списочное выражение. Рассмотрим оператор присваивания вида $\Sigma := A * B$;

Это оператор языка $\Delta\mathcal{E}$. Здесь A и B - имена исходных списков, Σ - имя списка, значение которого получается в результате выполнения списочной операции $*$ над исходными списками. Пусть исходные списки описаны на языке \mathcal{E} :

список $A [nA \times mA]$;

список $B [nB \times mB]$;

Например, список A состоит из однозначных восьмеричных чисел (разрядность таких чисел равна 3), а список B - из двузначных, при этом имеем следующие конкретные описания и конкретные значения этих списков:

список $A [2 \times 3]$; значение $A: (0,7)$.

список $B [3 \times 6]$; значение $B: (21,22,23)$.

Каждому оператору вида $\Sigma := A * B$, записанному на языке $\Delta\mathcal{E}$, можно поставить в соответствие его \mathcal{E} -образ, запись на языке \mathcal{E} .

\mathcal{E} -образ рассматриваемых списочных операторов присваивания будет иметь такую структуру. Сначала выписываются операторы вычисления длины $n\Sigma$ и разрядности $m\Sigma$ нового списка, а также описание этого списка

список $\Sigma [n\Sigma \times m\Sigma]$;

Затем описывается замкнутая процедура выборки элемента результирующего списка по номеру этого элемента. Наконец, строится оператор, вычисляющий значение нового списка многократным обращением к полученной процедуре $B\Sigma$. Этот последний оператор имеет одинаковый вид для всех списочных операторов присваивания, независимо от конкретной списочной операции, фигурирующей в выражении. Выпишем этот оператор:

начало $\kappa := 0$;

α : $\kappa := \kappa + 1$; $\kappa > n\Sigma$? да Ω ;

$B\Sigma(\kappa)$;

$\Sigma[\kappa] := t1$; на α ;

Ω : конец

Процедура выборки элемента результирующего списка целиком определяется списочной операцией.

Рассмотрим отдельно каждую из операций над списками.

I) Прямое произведение.

Списки рассматриваются как упорядоченные множества их элементов, и список Σ является прямым произведением двух множеств A и B в обычном понимании, т.е. каждая пара элементов $A[i]$ и $B[j]$ образует некоторый элемент нового списка. Значение элемента $\Sigma[k]$ составляется из значений соответствующих элементов $A[i]$ и $B[j]$, выписанных подряд как одно слово. Длина $n\Sigma$ нового списка и его разрядность $m\Sigma$ определяются соотношениями

$$n\Sigma = nA \times nB, \quad m\Sigma = mA + mB.$$

Номера элементов k, i, j связаны так:

$$k = 1, 2, \dots, n\Sigma; \quad i-1 < \frac{k}{nB} \leq i; \quad j = k - (i-1)nB.$$

В примере получается следующий список:

список $\Sigma [6 \times 9]$;

значение Σ : (021, 022, 023, 721, 722, 723).

Операции \otimes отвечает следующая процедура выборки:

замкнутая процедура $BBS.(k)$;

начало слово $S[mA:mB]$;

$ENT(k, nB)$;

$S[1] := A[i]$;

$i := i-1; i := i \times nB; i := k-i$;

$S[2] := B[i]$;

$t1 := S$; конец

Выбранный элемент является значением скаляра $t1$. Здесь процедура $ENT(a, b)$ вычисляет значение i - целое число, ближайшее сверху к величине a/b : $i-1 < a/b \leq i$.

2) Объединение списков

Список Σ получается объединением списков A и B в следующем порядке: сначала выписываются элементы списка A , затем элементы списка B ; при этом $n\Sigma = nA + nB$, $m\Sigma = \max(mA, mB)$.

$$\Sigma[k] = \begin{cases} A[k] & \text{при } k \leq nA, \\ B[i] & \text{при } i = k - nA > 0. \end{cases}$$

В примере получается следующий список:

список $\Sigma [5 \times 6]$;

значение $\Sigma : (*0, *7, 21, 23)$, здесь символом * обозначается любая восьмеричная цифра.

Операции объединения списков отвечает следующая процедура выборки:

замкнутая процедура $РБ\Sigma(k)$;

начало $k > nA$ да α ;

$t1 := A[k]$; на Ω ;

$\alpha : i := k - nA$; $t1 := B[i]$;

Ω : конец

3) Соединение списков

Значение элемента $\Sigma[k]$ составляется из значений элементов $A[k]$ и $B[k]$, взятых с тем же номером и выписанных подряд, как одно слово. При этом $m\Sigma = mA + mB$, $n\Sigma = \max(nA, nB)$. Если один из элементов $A[k]$ или $B[k]$ не имеет смысла (при $k > nA$ или $k > nB$), значение соответствующей части элемента $\Sigma[k]$ также является неопределенным (и обозначается символом *).

В примере получается следующий список:

список $\Sigma [3 \times 9]$;

значение $\Sigma : (021, 722, *23)$.

Операции соединения списков отвечает следующая процедура выборки:

замкнутая процедура $ББ\Sigma(k)$;

начало слово $S[mA : mB]$;

$S[1] := A[k]$;

$S[2] := B[k]$;

$t1 := S$; конец

3.2. Процедурные выражения

3.2.а. Синтаксис

<процедурное выражение> ::= <операция над процедурами>
(<последовательность имен процедур>)

3.2.в. Примеры

комп (A, B)
объед (A, B)
соед (A, B)
разв (A, B, P)
повт (A, B, P)

3.2.с. Семантика

Операции над процедурами определяются не на всем множестве процедур, а лишь на некотором его подмножестве, т.е. для процедур специального вида. А именно, рассматриваются процедуры, каждая из которых переводит некоторое исходное слово в другое слово или в список равнодлинных слов. Слова будем обозначать через x_A , x_B , x_P , y_P , x_T , а списки через ΣA , ΣB , ΣT . Как и ранее, под $n\Sigma A$ понимается длина списка ΣA . В частности, список может состоять и из единственного слова.

Предположим сначала, что исходные процедуры описаны на языке \mathcal{E} как открытые. Итак, имеем следующие описания:

открытая процедура $A(x_A, \Sigma A)$; < тело процедуры A > ;

открытая процедура $B(x_B, \Sigma B)$; < тело процедуры B > ;

При этом подразумевается, что процедура A переводит слово x_A в список ΣA , а процедура B переводит слово x_B в список ΣB .

Для выяснения смысла процедурных операций и выражений снова обратимся к оператору присваивания. В этом случае он имеет вид:

$$T := * (A, B)$$

или

$$T := ** (A, B, P)$$

Здесь $*$ обозначает одну из операций комп, объед или соед, а $**$ — одну из операций разв или повт. В результате выполнения этого оператора организуется новая процедура T , основные функции которой состоят в согласовании исходных процедур и управлении ими. Далее приводятся описания результирующих процедур, которые служат пояснением процедурных операций. Рассмотрим отдельно каждую из этих операций.

1) Композиция процедур

$$T := \text{комп} (A, B);$$

Композиция двух процедур A и B определяет собой новую процедуру T , которая состоит в том, что к исходному слову применяется сначала процедура A , а затем к каждому слову результата процедуры A применяется процедура B . Объединение всех результатов процедуры B , полученных в этом процессе, и является результатом новой процедуры T .

Процедуре T отвечает следующее описание:

открытая процедура $T(xT, \Sigma T)$;

начало $A(xT, \Sigma A)$; $\kappa := 0$;

α : $\kappa := \kappa + 1$; $\kappa > n \Sigma A$? да Ω ;

$B(\Sigma A[\kappa], \Sigma B)$; $\Sigma T := \Sigma T \cup \Sigma B$;

на α ;

Ω : конец

2) Объединение процедур

$T := \text{объед.}(A, B)$;

Объединение двух процедур A и B определяет новую процедуру T , которая состоит в том, что к исходному слову применяются поочередно обе процедуры, сначала процедура A , затем B . Объединение результатов этих процедур составляет результат новой процедуры T .

Процедуре T отвечает следующее описание:

открытая процедура $T(xT, \Sigma T)$;

начало $A(xT, \Sigma A)$; $B(xT, \Sigma B)$;

$\Sigma T := \Sigma A \cup \Sigma B$; конец

3) Соединение процедур

$T := \text{соед.}(A, B)$;

Соединение процедур отличается от предыдущей операции только тем, что над результатами процедур A и B выполняется операция соединения.

Процедуре T отвечает следующее описание:

открытая процедура $T(xT, \Sigma T)$;

начало $A(xT, \Sigma A)$; $B(xT, \Sigma B)$;

$\Sigma T := \Sigma A \mid \Sigma B$; конец

4) Разветвление процедур

$T := \text{разв.}(A, B, P)$;

В этой и следующей операциях в число аргументов добавляется специальная открытая процедура $P(xP, yP)$, переводящая каждое слово xP в единственное слово yP .

Разветвление двух процедур A и B с распознающей процедурой P определяет новую процедуру T , которая при $yP=0$ совпадает с процедурой A , а при $yP \neq 0$ - с процедурой B . P является специальной распознающей процедурой, вырабатывающей условие.

Процедуре T отвечает следующее описание:

открытая процедура $T(xT, \Sigma T)$;

начало $P(xT, yP)$; $yP=0$? да α ;

$B(xT, \Sigma T)$; на Ω ;

α : $A(xT, \Sigma T)$;

Ω : конец

5) Повторение процедуры

$T := \text{повт } (A, B, P)$;

В операции повторения, основной рабочей процедурой является процедура A , многократно применяемая к обновляемому значению исходного слова. Процедура B перевычисляет новое значение исходного слова. P является распознающей процедурой и вырабатывает условие, останавливающее процесс.

Процедуре T отвечает следующее описание:

открытая процедура $T(xT, \Sigma T)$;

начало

α : $P(xT, yP)$; $yP=0$? да Ω ;

$A(xT, \Sigma A)$; $\Sigma T := \Sigma T \cup \Sigma A$;

$B(xT, xT)$; на α ;

Ω : конец

В случае, когда исходные процедуры являются замкнутыми, такими же являются и результирующие процедуры. При этом их смысл полностью согласуется с приведенным выше истолкованием. Отметим, что представляется удобным трактовать и исходные и результирующие процедуры как замкнутые процедуры вычисления и выборки очередного результирующего слова. Здесь имеется известная аналогия с реализацией списочных операций (3.1.с).

§ 2. Новые конструкции языка

4. Описание схемы

4.а. Синтаксис

$\langle \text{описание схемы} \rangle ::= \text{схема} \quad \langle \text{тело схемы} \rangle ;$
 $\langle \text{тело схемы} \rangle ::= \langle \text{последовательность строк} \rangle$
 $\langle \text{посл. строк} \rangle ::= \langle \text{строка} \rangle \mid \langle \text{посл. строк} \rangle , \langle \text{строка} \rangle$
 $\langle \text{строка} \rangle ::= \langle \text{левая часть строки} \rangle = \langle \text{правая часть строки} \rangle$
 $\langle \text{левая ч. строки} \rangle ::= \langle \text{объект} \rangle$
 $\langle \text{правая ч. строки} \rangle ::= \langle \text{выражение} \rangle$

4.в. Пример

схема $\alpha = \beta + c,$
 $\beta = d \times c,$
 $d = e - f;$

4.с. Семантика

По существу схема является описанием сложного выражения. Схема описывает объект, имя которого стоит в левой части первой строки. В приведенном выше примере 4.в схема является описанием объекта α . В остальном порядок строк в схеме безразличен. Объекты, которые стоят в правых частях схемы, либо встречаются в левых частях строк той же схемы (таковы объекты β и d в нашем примере), либо являются опорными, т.е. должны быть описаны на \mathcal{E} -языке (таковы объекты c, e , и f в том же примере). В частности, эти опорные объекты могут быть скалярами и не требовать никакого другого описания.

Схема задает бинарное отношение старшинства в множестве объектов, имена которых фигурируют в схеме. А именно: каждый объект, стоящий в левой части строки, является старшим по отношению к объектам, стоящим в правой части той же строки. Будем обозначать отношение старшинства символом (\gg). Тогда схема из примера 4.в определяет следующее отношение:

$$\alpha \gg \beta, \alpha \gg c, \beta \gg d, d \gg e, d \gg f.$$

Каждое отношение вида $\alpha \gg \beta$ можно интерпретировать так: значение объекта β необходимо непосредственно для вычисления значения объекта α .

Граф указанного отношения старшинства для схемы из 4.в изображен на рисунке 1.

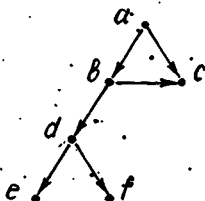


Рис. 1.

Объектам схемы отвечают вершины графа. Дуги отражают отношение старшинства. Каждому отношению $a \gg b$ отвечает дуга (a, b) , направленная из вершины a в вершину b .

Отношение, обратное старшинству, будем называть подчиненностью. Так, отношение $b \ll a$ означает, что b подчиняется a . Граф отношения подчиненности для нашей схемы изображен на рисунке 2.

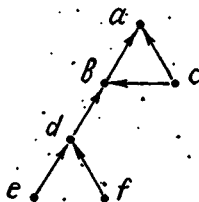


Рис. 2.

Графы отношений подчиненности и старшинства отличаются только ориентацией дуг (направлением стрелок), они получаются один из другого изменением ориентации каждой дуги на противоположную. Поэтому безразлично, какой из этих графов рассматривать. Для определенности остановимся на графе подчиненности.

Наряду с основным бинарным отношением, порождающим данный граф (в нашем случае это отношение подчиненности), в графе рассматривается новое бинарное отношение, а именно: вершина x предшествует вершине y ($x \leq y$), если в графе существует путь из x в y . В нашем примере $f \leq b$, $e \leq a$, но $c \neq d$ и т.п.

Отношение предшествования является транзитивным замыканием

исходного отношения подчиненности. Отношение $f \leq b$ в нашей схеме означает, что значение объекта f должно быть вычислено раньше, чем значение объекта b (т.е. значение объекта f используется в процессе вычисления значения объекта b - непосредственно или косвенно).

Замкнутый путь, начинающийся и заканчивающийся в одной и той же вершине, называется контуром. В языке запрещаются схемы, которым отвечают графы с контурами. Формально это означает, что в графах допустимых схем отношение предшествования является асимметричным, т.е.

$$x \leq y, y \leq x \rightarrow x = y.$$

Таким образом, в допустимых схемах транзитивное замыкание исходного отношения подчиненности является отношением порядка, и множество объектов допустимой схемы частично упорядочено этим отношением. Другими словами, в языке допускаются только такие схемы, для которых вычисление значений объектов может быть упорядочено во времени без нарушения отношения подчиненности. Отметим также, что допустимым схемам отвечают графы подчиненности с единственной конечной вершиной.

5. Описание сложного списка

5.а. Синтаксис

<описание сложного списка> ::= список-схема <последов. списочных строк> ;

<списочная строка> ::= <имя списка> = <списочное выражение>

5.в. Пример

список-схема $\Sigma = A \otimes B,$
 $A = C \parallel D,$
 $B = E \cup F;$

5.с. Семантика

Имя описываемого сложного списка стоит в левой части первой строки схемы: в примере 5.в. описан список Σ . На рисунке 3 изображен граф старшинства для этого списка.

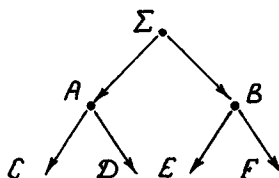


Рис. 3.

Списки C , D , E , F являются опорными, списки A и B описаны в этой же схеме. Значение сложного списка должно строиться из значений опорных списков в соответствии со смыслом операций над списками (3.1). Программным образом описания сложного списка является процедура выборки элемента сложного списка по номеру этого элемента.

Пусть заданы следующие описания и значения опорных списков (значения списков заключены в круглые скобки, значения элементов отделяются запятыми):

список C $[2 \times 3]$; значение C : (0, 7);

список D $[2 \times 6]$; значение D : (11, 12);

список E $[1 \times 6]$; значение E : (20);

список F $[2 \times 6]$; значение F : (33, 34).

Схема определяет следующие значения промежуточных списков A и B и результирующего списка Σ :

$A = C \parallel D$: (011, 712);

$B = E \cup F$: (20, 33, 34);

$\Sigma = A \otimes B$ (01120, 01133, 01134, 71220, 71233, 71234).

Спецификой списка-схемы является то, что он является "динамическим" объектом, его значение фактически не выписывается целиком и не хранится в машинной памяти. Вместо этого по схеме конструируется процедура выборки нужного элемента этого списка.

6. Описание сложной процедуры

6.а. Синтаксис

$\langle \text{описание сложной процедуры} \rangle ::= \underline{\text{процедура-схема}} \quad \langle \text{последовательность процедурных строк} \rangle$

$\langle \text{процедурная строка} \rangle ::= \langle \text{имя процедуры} \rangle = \langle \text{процедурное выражение} \rangle$

6.в. Пример

процедура-схема $T = \text{разв} (A, B, P),$
 $A = \text{комн} (C, D),$
 $D = \text{объед} (E, F),$
 $F = \text{сгед} (K, L),$
 $C = \text{ловт} (N, M, Q);$

6.с. Семантика

В примере 6.в описана сложная процедура T . На рисунке 4 изображен граф старшинства для этой процедуры.

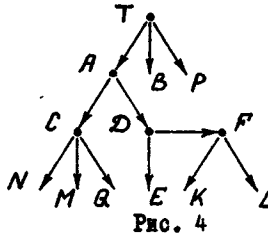


Рис. 4

Программный образ описания сложной процедуры строится из образов подчиненных процедур в соответствии со смыслом операций над процедурами (3.2). Процедура T переводит каждое исходное слово в одно или несколько результативных слов. При каждом обращении к этой процедуре выдается либо очередное результативное слово, либо сигнал об исчерпании результата.

7. Укрупненный оператор процедуры

7.а. Синтаксис

$\langle \text{укрупненный оператор процедуры} \rangle ::= \text{оператор} \langle \text{имя процедуры} \rangle (\langle \text{имя списка } \Delta \mathcal{E} \rangle, \langle \text{последовательность имен списков } \mathcal{E} \rangle);$

7.в. Пример

оператор $T(A, B, C);$

7.с. Семантика.

Укрупненный оператор процедуры реализует специальный режим обработки и вычисления списков. В этом операторе фигурируют и в полной мере используются введенные в языке $\Delta \mathcal{E}$ укрупненные объекты: списки-схемы и процедуры-схемы.

В примере 7.в процедура T может быть простой или сложной. Первый параметр процедуры (в примере это список A) является ее аргументом и может быть простым или сложным списком. Остальные параметры являются простыми списками, это результаты процедуры (в примере списки B и C).

Выполнение оператора состоит в применении процедуры последовательно к каждому элементу списка - аргумента. Результативные элементы процедуры накапливаются в простых результативных списках.

Можно отметить следующие особенности укрупненного оператора процедуры. Этот оператор описывает массовые вычисления: однотипным преобразованиям подвергается каждый элемент исходного (сложного) списка. При этом здесь явно выделены и зафиксированы две стороны вычислительного процесса:

1) одготовка и формирование очередной порции исходной информации,

2) собственно вычислительный алгоритм.

Для описания этих двух сторон процесса здесь используются иерархические схемные структуры (список-схема и процедура-схема). Представляется целесообразным выделить в языке по возможности эти стороны вычислительного процесса - и потому, что для описания каждой из них используются свои специфические средства, и потому также, что в принципе они могут реализоваться параллельно.

Л и т е р а т у р а

1. Канторович Л.В. Перспективы работы в области автоматизации программирования на базе крупноблочной системы. - "Труды МИ АН СССР", 1968, 96, с.5-15.
2. Канторович Л.В. О проведении численных и аналитических вычислений на машинах с программным управлением. - "Известия АН Арм.ССР", 1957, 10, №2, с.3-16.
3. Петрова Л.Т. Некоторые применения схемной символики. - "Журнал выч. мат. и мат. физики", 1961, 1, №3, с.513-522.
4. Петрова Л.Т., Платунова И.А. Реализация на машине вычислений в исходном классе списков. - "Труды МИ АН СССР", 1961, 66, с.16-36.
5. Поттосин И.В., Рар А.Ф., Катков В.Л. ЭПСИЛОН-система автоматизации программирования для задач символьной обработки. - "Труды I Всесоюзной конф. по программированию", Киев, 1968, серия Л, с.88-108.
6. Марков А.А. Теория алгоритмов. - "Труды МИ АН СССР", 1954, 62.

Поступила в редакцию
28.П. 1972 г.