УДК 004.424, 004.415

В.Г. Колесник

DS-ТЕОРИЯ. ПРЕДСТАВЛЕНИЕ КАНОНИЧЕСКОГО АЛГОРИТМА С ПОМОЩЬЮ АЛГОРИТМИЧЕСКОГО ЯЗЫКА

Работа является продолжением описания схемы декомпозиции как теоретической модели, обеспечивающая возможность генерации прикладных алгоритмов. Приведено описание алгоритмического языка, предназначенного продемонстрировать возможность генерации алгоритмов. Описана одна из групп факторов – способы размещения свойств на ленте абстрактного типа, учет которой, позволяет превратить канонический алгоритм в реальный прикладной алгоритм или, что то же, превратить схему декомпозиции в текст программы. Вводятся понятия алгоритмического примитива и алгоритмического сочленения (операнда и операции), как средств построения алгоритма. Данные понятия и способ построения алгоритмов – это альтернативная понятийного аппарата и методологии структурного программирования. Вводятся понятия функционального ядра, алгоритмического фрейма, функционального содержания и алгоритмической матрицы.

Введение

В работе [1] приводится описание теоретической модели, называемой схемой декомпозиции. Там же показано, что схема декомпозиции (DS) имеет алгоритмическую природу. Здесь описаны контур синтеза (КС) и канонический (универсальный) алгоритм (КА) как основные атрибуты схемы декомпозиции.

DS-теория [1], неотъемлемой частью которой является DS, как теоретическая модель, создает предпосылки для генерации алгоритмов (не машинного кода). Несмотря на то, что КС и КА вполне формализованы, для того, чтобы они были реализованы в практической обработке как программа для компьютера, их необходимо представить средствами алгоритмического строчного языка. В настоящей статье вводится описание алгоритмического языка. Язык предназначен не для практического программирования, а только для того, чтобы продемонстрировать возможность генерации алгоритмов и текст сгенерированной программы.

В работе вводится понятие алгоритмического примитива, использующийся как компонента при построении DS. Описываются свойства алгоритмических примитивов. Приводятся конструкции операторов и предложений на алгоритмическом языке, с помощью которых представляются алгоритмические примитивы. Описываются понятия алгоритмических сочленений как операций над алгоритмическими объектами.

КА, который может быть построен на основе DS, на пути его преобразования в реальную программу, должен быть доработан с учетом конкретной вычислительной среды, в которой будет функционировать программа. Те факторы (алгоритмически релевантные факторы), которые необходимо учесть, – многообразны и их достаточно много. С точки зрения *DS*-теории, они собраны в группы. В статье описывается процедура внесения изменений в КА, порождаемых одной группой факторов – способам размещения P-данных на A-ленте.

Несмотря на то, что алгоритм как результат генерации, представляется с помощью алгоритмического языка императивного типа, DS, как описание исходное для генерации, является описанием декларативного типа. Сгенерированный алгоритм по определению отражает порядок вычислений, а описание DS явно этот порядок не показывает. В процессе генерации алгоритма учитывается порядок обхода дерева полной схемы декомпозиции (DPS) сверху вниз и слева направо и благодаря этому он появляется в сгенерированном алгоритме. По сути, DS – это объект декларативного программирования, не

имеющий внешнего сходства с традиционными языками декларативного программирования [2, 3].

Алгоритмические компоненты DS

Далее используется понятие примитив схемы декомпозиции (ПСД). Простейшими элементами, или примитивами, из которых строится схема декомпозиции являются аналитическая алгоритмическая зависимость (ААЗ), синтетические зависимости (САЗ): первого ($САЗ^1$) и второго вида ($САЗ^2$). Также как ПСД рассматривается механизм декомпозиции (DM). Из ПСД строятся более сложные конструкции. Прототипы ПСД – это базовые конструкции структурного программирования [4].

САЗ состоит из двух частей: заголовка и тела. Заголовок определяет условия начала и условия завершения работы САЗ, а тело выполняет работу с очередной компонентой С-свойства [1]. Пусть два узла связаны ветвью (родственной связью). С ними соотносится некая САЗ. Если САЗ соотносится с узлом родителем, то в списке А-зависимостей, связанных с этим узлом, будет выполняться заголовок САЗ, а тело САЗ будет выполняться в списке А-зависимостей связанных с узлом потомком.

Механизм декомпозиции представляется набором предложений DM = (S_{begin}, S_f, S_{next}, S_{end}), где S_{begin} – условие начала декомпозиции, S_f – описание процедуры первого шага декомпозиции, – описание работы с первой частью, S_{next} – описание процедуры очередного шага декомпозиции, S_{end} – описание условия завершения декомпозиции. S_{begin}, S_f, S_{end} – эти три компоненты называются заголовком механизма декомпозиции. S_{next} – эта компонента называется телом механизма декомпозиции.

ААЗ, заголовок САЗ и DM, тело САЗ и DM называются простейшими расчетными процедурами (ПРП).

С листьями и узлами DPS может соотноситься произвольное количество А-зависимостей. Если результат реализации одной А-зависимости исходное данное для реализации другой А-зависимости, то А-зависимости между собой зависимы. При записи таких А-зависимостей должен сохраняться как порядок их перечисления, так и порядок их выполнения. Если порядок реализации А-зависимостей произвольный, то А-зависимости между собой независимы.

Возможна ситуация когда расчетная операция, реализующая А-зависимость может быть выполнена только при определенном условии. Это связано с тем, что существует условный тип части объекта, в момент декомпозиции может отсутствовать некоторое Р-свойство. Из-за этого в КС в момент расчета не будет рассчитываться (реализовываться) некоторая А-зависимость. Такие А-зависимости называются условными.

Две условные А-зависимости могут быть связаны. Суть их связи заключается в том, что расчетные процедуры, что их реализуют, выполняются в зависимости от определенного значения некоторого Р-свойства. Такого рода условных А-зависимостей может быть и более двух. Они называются альтернативными.

С узлом DPS, который не является конечным, может быть соотнесено произвольное количество ААЗ, заголовков DM, заголовков собственных САЗ. С узлом, который не является корневым, может быть соотнесено произвольное количество ААЗ, тело САЗ и одно тело DM.

Учитывая то, что с каждым из листьев может быть связано произвольное (большое) количество ААЗ, компонентов САЗ и компонентов DM, необходимо выполнить предварительную обработку всего набора ПРП. Для этого надо выполнить следующее.

1. Собрать и объединить зависимые ПРП в порядке их взаимной зависимости. После этого обозначить эту алгоритмическую конструкцию (АК) как единое целое.

2. Собрать и скомпоновать условные ПРП. Принцип объединения в том, что у них одинаковое или различные значение одного и того же Эл-свойства. В результате тоже формируется АК, которая должна быть обозначена как единое целое.

3. Собрать и объединить альтерна-

тивные ПРП. В результате тоже формируется АК, которая должна быть обозначена как единое целое.

АК и ПРП также как и А-зависимости тоже могут быть условными и безусловными, а также зависимыми или независимыми.

Предыдущие два шага повторяются несколько раз. Но анализу и объединению как компоненты подвергаются и ПРП, и АК. Из них могут быть сформированы АК более сложные. Анализ и объединение завершится тогда, когда в списке расчетных процедур будут перечислены все, связанные с одним узлом DPS и ПРП, и сложные АК, которые независимы друг от друга. Перечень этих независимых ПРП и АК называется алгоритмической конструкцией узла (листа) – (АКУ). На DPS на каждом узле и листе будет сформирована собственная АКУ.

Более точное определение КС звучит так: контур синтеза – это перечень АКУ, составленный в процессе обхода DPS сверху вниз и слева направо. Составление КС как перечня АКУ – это первый шаг синтеза конечного алгоритма, который пригоден для обработки в компьютере.

Р-свойства и Р-данные

В работе [1] предложены определения Эл-свойства, А-свойства, С-свойства и их родового понятия – Р-свойства.

С-свойство, состоящее из одного Эл-свойства, называется простым. А-свойство, состоящее из произвольного количества Эл-свойств, называется простым. С-свойство, состоящее из простого А-свойства, называется расширенным. А-свойство, в состав которого входит одно или более простое или расширенное С-свойство, называется расширенным.

С-свойство, состоящее из расширенного А-свойства, называется сложным. С каждым типом части объекта или самого объекта соотносится только одно А-свойство. Это А-свойство называется главным А-свойством для данного типа части объекта. Главное А-свойство может состоять из промежуточных А-свойств, входящих в главное как компоненты. Промежуточные могут содержать А-свойств, которые тоже промежуточные. То есть, существует иерархическая структура А-свойств.

Р-свойства могут фиксироваться на носителях, доступных для машинной обработки. В связи с этим наряду со свойствами объекта рассматриваются также данные об объекте. Данное об объекте (Р-данное) или данная величина, характеризующая объект, – это свойство объекта, для которого определен знак, читаемый или воспринимаемый человеком или (и) техническими средствами. Аналогично понятиям С-свойство, Эл-свойство и А-свойство существуют понятия: совокупное данное (С-данное), агрегированное данное (А-данное) и элементарное данное (Эл-данное) и их родовое понятие – Р-данное. Р-свойства и Р-данные соотносятся как с объектом, так и с частями объекта. А-данные могут включать наряду с Эл-данными А-данные и С-данные. С-данные могут содержать А-данные. Рассматриваются также расширенные С-данные и А-данные, сложные С-данные и А-данные и главное А-данное.

Каждая часть объекта должна быть поименована. Эл-данное, что содержит идентификатор объекта или части объекта, обязательно должно присутствовать в составе главного А-данного. В составе А-данного этот идентификатор называется ключом. Ключ – это обобщенное понятие традиционно понимаемого в обработке файлов и баз данных ключа записи. Конкретные числовые, текстовые и прочие значения Р-свойств и Р-данных называются Д-данными. Прототип понятия Д-данное в электронной обработке данных – просто данные.

Носитель Д-данных

Рассматриваются носители Д-данных двух видов: А-память и А-лента. А-память – это обобщенное абстрактное понятие оперативной памяти компьютера. Доступ к Д-данным размещенным в А-памяти реализуется упоминанием имени Р-данного. Понятие “А-память” до предела упрощено и предназначено для моделирования алгоритмов реализующих А-зависимости.

А-лента – это абстрактный образ ленты (магнитной или бумажной), состоящей из ячеек, вдоль которой движется читающее устройство (А-головка). В этой статье понятие А-ленты усложняется по сравнению с [1]. Допускается, что ячейки А-ленты группируются в записи. Размер записи определяется размером простого А-данного, которое содержится в записи. А-лента может быть разделена на участки, которые больше записей. То есть, существует иерархическая структура вложенных участков А-ленты. Эти участки называются областями и подобластями.

При движении А-головки вперед выполняется чтение, а при движении назад выполняется только перемотка. А-головка может считывать записи с А-ленты и передавать в А-память или выбирать из А-памяти и записывать записи на А-ленту. После того как запись считана в А-память, становиться возможным доступ к Д-данным. Доступ к записям на А-ленте только последовательный. То есть, для того чтобы прочитать десятую запись, необходимо перед этим прочитать девять записей предшествующих десятой. С помощью А-ленты моделируется хранение Д-данных и иллюстрируется процесс генерации алгоритмов.

Типы записей, областей и подобластей на А-ленте должны быть различимы. Типы области и подобласти А-ленты именуются. Например, большие латинские буквы с нижними индексами (или без).

Тот факт, что область или подобласть некоторого типа содержит произвольное количество записей или подобластей будет отмечаться знаком μ на уровне верхнего индекса рядом с названием типа области. Или R^{μ}=<R1>, что означает “Область типа R (или область R) содержит произвольное количество подобластей типа R1”.

Тот факт, что область или подобласть содержит определенное количество записей или подобластей будет отмечаться знаком ν на уровне верхнего индекса рядом с названием области. R^{ν}=<R2,R3,R4>, что означает “область R содержит три подобласти R2, R3 и R4. Каждая из них встречается один раз”. Порядок перечисления подобластей в этом предложении соответствует порядку их размещения в области. Если это не так, то иной порядок оговаривается.

Для описания того, как расположены и вложены друг в друга области и подобласти А-ленты используются предложения описанных двух типов. Все предложения вместе составляют кортеж RG, с помощью которого описывается структура областей и подобластей на А-ленте. Размещение может быть представлено в виде дерева.

Размещение Р-данных на А-ленте

Существует несколько видов размещения Р-данных в областях и подобластях.

Простое А-данное. Наименьшая неделимая часть А-ленты – это запись. В записи размещается простое А-данное. То, что А-данное Q размещено в записи R, записывается следующим образом R[Q].

Записи и подобласти делят А-ленту на фрагменты. Далее эти фрагменты называются А-фрагментами.

Расширенное С-данное. Расширенное С-данное занимает область или подобласть, состоящую из записей. Одна запись содержит одно А-данное, являющееся компонентой С-данного. Область или подобласть, занятая расширенным С-данным, называется простой.

Пример: Объект имеет совокупное свойство RA, которое состоит из Эл-свойства AD (рис. 1). Область D содержит

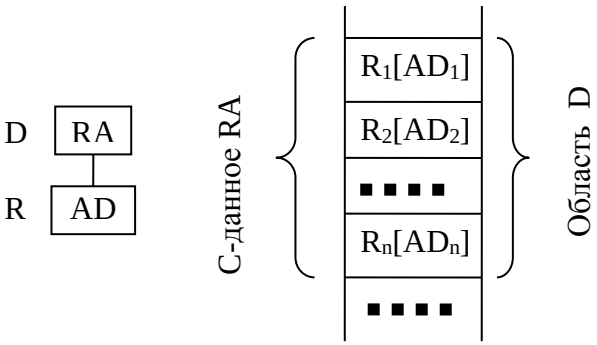


Рис. 1. Размещение С-данного RA в области D на А-ленте

произвольное количество записей R. Элементарное свойство AD размещено в записи R. Таким образом совокупное свойство RA размещается в области D.

FS: RA = {∪AD}.

RG: D [RA], R [AD], D^{μ} = <R>.

Так как для Р-свойств RA и AD определено размещение, то они одновременно являются и Р-данными. Хотя после размещения Р-свойств их имена можно не присваивать соответствующим Р-данным, но присвоить другие имена.

А-фрагменты должна быть различимы на А-ленте. Для А-фрагментов должны быть определены признаки начала и конца.

Признаком начала подобласти может быть:

- служебная запись – маркер;
- некое служебное Эл-данное (тип записи), содержащееся в каждой записи С-данного. Это Эл-данное имеет определенное значение. Если в записях хотя бы одного С-данного есть служебное Эл-данное, то оно должно быть во всех записях компонентах исходного расширенного А-данного;
- изменение значения ключа. То есть, во всех записях, компонентах исходного А-данного есть Эл-данное с одним и тем же значением. Оно характерно тем, что для одного С-данного, это Эл-данное имеет одно и то же значение.

Признаком конца подобласти может быть:

- служебная запись – маркер;
- изменение значения служебного Эл-данного (типа записи);
- изменение значения ключа;
- конец А-ленты.

Способы размещения А-фрагментов

Есть три способа размещения А-фрагментов внутри области А-ленты.

Порядок относительно границ области размещения (ORb). Всем А-фрагментам содержащим компоненты расширенного А-данного установлен номер следования относительно начала или (и) конца области (рис. 2).

Пример:

FS: Z = {A,B,C,D,E}. A, B, D – простые А-данные. С и Е – простые С-данные.

RG: R^{v}=<R1[A],R2[B],R3[C],R4[D], R5[E]>, $R3^{\mu}$=< R31>, $R5^{\mu}$ =<R51>. Здесь R1, R2, R4, R31, R51 – записи, R3, R5 – подобласти.

R1 и R2 могут иметь или не иметь тип записи.

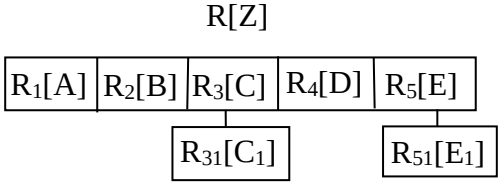


Рис. 2. Компоненты расширенного А-данного имеют номера следования относительно начала области

Все остальные записи должны иметь тип записи обязательно. Если нет типа записи, то должно быть Эл-данное выполняющее функции ключа и для записей R4, R31 и R51 должны быть определены значения ключа.

Для всех А-фрагментов установлен номера следования в зоне А-ленты: R1 – первый, R2 – второй, R3 – третий, R4 – четвертый, R5 – пятый. Начало подобласти R3 первая запись R31, конец подобласти R3 –запись R4. Начало подобласти R5 первая запись R51, конец подобласти R5 – конец А-ленты.

Относительный порядок (ORn). Два или более А-фрагмента имеют порядок следования относительно друг друга.

Пример:

FS: Z = {A,B,C,D,E}. A, B, D – простые А-данные. С и Е – простые С-данные.

RG: R^{v}=<R1[A],R2[B],R3[C],R4[D], R5[E]>, $R3^{\mu}$=< R31>, $R5^{\mu}$ =<R51>.

Здесь R1, R2, R4, R31, R51 – записи, R3, R5 – подобласти.

Подобласть R3 следует непосредственно после записи R2. Записи R31 и R51 размещены компактно и должны иметь тип записи или ключ с определен-

ным значением. Начало подобласти R5 первая запись R51, конец подобласти R5 – любая запись отличающаяся от R51. Если подобласть R5 в конце А-ленты, то признак конца подобласти будет конец А-ленты. Начало подобласти R3 – запись, которая непосредственно следует после записи R2. Конец подобласти R3 – любая запись отличающаяся от R31. Если подобласть R3 в конце А-ленты, то признак конца подобласти будет конец А-ленты.

Возможен способ размещения, когда два А-фрагмента не следуют друг за другом непосредственно. Такая ситуация может возникнуть, когда А-фрагменты, не участвуют при обработке и их необходимо пропускать при чтении А-данных с А-ленты. Это так называемые непродуктивные Д-данные.

Порядок по значению ключа (ORs). Простая подобласть, содержащая А-фрагменты может иметь характеристику – порядок своих компонент. Записи могут быть упорядочены по значениям одного или нескольких Эл-данных входящих в состав А-данного содержащегося в записи. Записи могут быть упорядочены по возрастанию и по убыванию ключа. Могут быть и более сложные отношения между ключами записей. Порядок между записями – это характеристика подобласти.

Размещение расширенных А-данных может быть многообразным. Причины этому следующие:

- расширенное А-данное может состоять из простых А-данных. Последние размещаются в записях и вперемешку с расширенными С-данными. Так называемые промежуточные А-данные;
- количество промежуточных А-данных может быть произвольное;
- количество промежуточных С-данных может быть произвольное.

Совмещение различных способов размещения А-фрагментов

Компоненты расширенного А-данного при размещении на А-ленте могут иметь различный порядок. Существует несколько способов совместного размещения компонент.

Совместное размещение первого вида (mix1). Для А-фрагментов порядок размещения не установлен.

FS: Z = {A,B,C,D,E}. A, B, D – простые А-данные. C и E – простые С-данные.

RG: R^{v} = <R1[A], R2[B], R3[C], R4[D], R5[E]>, $R^{\mu}3$=< R31>, $R^{\mu}5$ = <R51>, R1, R2, R4, R31, R51 – записи; R3, R5 – подобласти.

Ситуация показана на рис. 3. Все записи должны иметь тип записи. Каждая из подобластей R3 и R5 размещается компактно. Начало и конец каждой из подобластей должен быть определен.

Начало области R одна из записей R1, R2, R4 – или начало подобластей R3 (запись R31), R5 (запись R51). Конец R это конец А-ленты или запись, отличающаяся от R1, R2, R4, R31, R51.

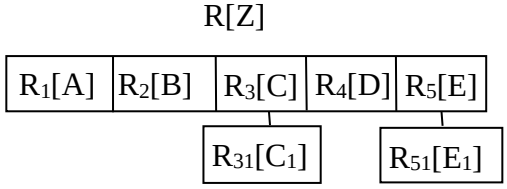


Рис.3. Компоненты расширенного А-данного имеют совместное размещение

Начало R3 – запись R31, конец R3 – запись, отличающаяся от R31. Начало R5 – запись R51, конец R5 – запись, отличающаяся от R51.

Записывается это так:

RG: R=<R1[A]° R2[B]° R3[C]° R4[D]° R5[E]°>; $R3^{\mu}$ =< R31>; $R5^{\mu}$ = < < R51>. Здесь R1, R2, R4, R31, R51 – записи; R3, R5 – подобласти. Знак ° рядом с именами записей или подобластей показывает, что эти записи или подобласти перемешаны.

Совместное размещение второго вида (mix2). Некоторые А-фрагменты могут иметь относительный порядок следования, а остальные имеют произвольный порядок. В этом случае выполняется размещение в так называемых рабочих по-

добластях. Выполняется группировка компонент. Если какие-либо компоненты имеют порядок относительно друг друга и размещены рядом, то они объединяются в подобласть. В результате рабочие подобласти и остальные компоненты имеют совместное размещение mix1.

FS: Z = {A,B,C,D,E}. A, B, D – простые A-данные. C и E – простые C-данные.

RG: R^{v}=<R1[A]° R6° R4[D]° R5[E]°>, $R3^{\mu}$ = < R31>, $R5^{\mu}$ = <R51>, здесь R1, R2, R4, R31, R51 – записи; R3, R5 – подобласти. R6 = <R2[B], R3[C]> – рабочая подобласть.

Совместное размещение третьего вида (mix3). Одновременно некоторые A-фрагменты имеют определенное место относительно границ области размещения, а другие могут быть перемешаны. R1 – первый, R2 – последний, а запись R4 и подобласти R3 и R5 перемешаны в одной области. Записи R31, R4 и R51 должны иметь тип. Признак начала подобласти, где они хранятся – первая встретившаяся среди R31, R4 и R51. Признак конца подобласти – конец A-ленты или запись отличная от R31, R4 и R51. Если у некоторых подобластей или записей есть порядок относительно границ области, то они могут быть в начале или в конце содержащей их области.

FS: Z = {A,B,C,D,E}. A, B, D – простые A-данные. C и E – простые C-данные.

RG: R^{v}=<R1[A], R3[C]° R4[D]° R5[E]° R2[B] >, Rμ3=< R31>, $R^{\mu}5$ =<R51>, здесь R1, R2, R4, R31, R51 – записи; R3, R5 – подобласти.

Совместное размещение четвертого вида (mix4). A-фрагменты могут иметь относительный порядок следования, порядок относительно границ области или (и) произвольный порядок.

FS: Z = {A,B,C,D,E,F,G}. A, B, D, F,G – простые A-данные. C и E – простые C-данные.

RG: R^{v}=<R1[A], R2[B], R3[C], R4[D], R5[E] , R6[F], R7[G] >, $R3^{\mu}$ = < < R31>, $R5^{\mu}$ = <R51>. R1, R2, R4, R31, R51, R6, R7 – записи; R3, R5 – подобласти.

R1 – первая в области – R, R6 – предпоследняя и R7 – последняя. R4 и R5 – следуют непосредственно друг за другом.

В этом случае выполняется размещение в рабочих подобластях. Выполняется группировка A-фрагментов. Если какие-либо A-фрагменты имеют порядок относительно друг друга и размещены рядом, то они объединяются в подобласть. В случае конкретного примера этот вид размещения записывается следующим образом:

RG: R^{v}=<R1[A], R2[B]° R3[C]° R8°, R6[F], R7[G] >, $R3^{\mu}$ = < R31>, $R5^{\mu}$ = <R51>, здесь R1, R2, R4, R31, R51 , R6, R7 – записи; R3, R5 – подобласти. R8 = < R4[D], R5[E] > – рабочая подобласть.

Входная и выходная A-ленты

Существование A-зависимостей и КС предполагает, что есть P-свойства (соответственно и P-данные) исходные для расчетов и P-свойства являющиеся результатом в расчетах. Если реализуется ААЗ, то и операнды, и результат размещены в A-памяти, а затем выводятся в одной и той же записи. Но возможна ситуация, что перед расчетом P-данные размещены на одной A-ленте, а результат расчета должен быть выведен на другую A-ленту. В DS-теории предусматривается возможность работы с P-данными размещенными во входных и выходных A-лентах. A-лента, используемая для хранения исходных для расчета P-данных, называется входной. A-лента, используемая для хранения P-данных, являющихся результатом в расчетах, называется выходной.

Представление ПСД средствами алгоритмического языка

Из-за того, что размещение A-фрагменты, содержащие P-данные, имеет различные вариации, описанные выше, КА – более сложная конструкция по сравнению с описанной в [1]. Далее КА будет описываться с помощью алгоритмического языка подобного ранним версиям языка КОБОЛ. Для описания алгоритмов используется алгоритмический язык, синтаксические конструкции, которого вводятся по мере их необходимости. Язык не имеет строго формального определения, а осно-

ван на интуитивном понимании традиционных строчных языков. По ходу изложения определения могут быть уточнены на минимально необходимом уровне. Назначение языка временное.

Реализация ААЗ. Наименьшая и основная частью КС – это действие, реализующее одну ААЗ. Оно описывается оператором присваивания. Точнее, оператором присваивания, существующем в традиционных алгоритмических языках в их простейшей форме.

Эл-свойства А и В связанные знаком равно А = α (В) реализуются в виде оператора присваивания А:=В, который обозначает: "На основании Эл-свойства В, в соответствии с существующей между А и В зависимостью α, формируется Эл-свойство А". Зависимость α может содержать традиционно используемые знаки арифметических и алгебраических операций, разделителей. Если по ходу изложения будет необходимость использовать алгебраические функции, которые в традиционных алгоритмических языках реализуются подпрограммами, то это будет оговариваться особо. Также будет особо оговариваться ситуация, когда ААЗ существует, но не может быть выражена композицией арифметических или (и) алгебраических знаков. Частный случай оператора присваивания – это пересылка значения из В в А.

Операторы, порождаемые ААЗ, тоже могут быть зависимыми или не зависимыми.

Алгоритмические сочленения. Далее описываются способы объединения операторов алгоритмического языка в более сложные конструкции. Объединение выполняется с помощью, так называемых алгоритмических сочленений – последовательного, условного и альтернативного. Сочленения – это операции над алгоритмическими объектами (и, соответственно, над операторами и предложениями алгоритмического языка).

Группа операторов называется предложением. Предложение завершается точкой или ограничивается операторными скобками, которые будут описаны далее. Предложение может быть разделено на части только с точки зрения удобства зрительно восприятия. Внутри предложения операторы разделяются запятой или пробелом (пробелами).

Операторы в предложении объединены **последовательным** сочленением. Предложение далее рассматривается как неделимая конструкция, и может быть использовано как компонента в дальнейших сочленениях, в результате чего получаются сложные предложения. Предложение закрепляет порядок выполнения операторов и, соответственно, ААЗ, независимо от того, какими они были перед компоновкой – зависимыми или независимыми.

Условные ААЗ или ПРП представляются условными операторами – (IF THEN). При описании условий используются традиционные знаки сравнения (>, <, =, ≥, ≤, ¬), логические операции (OR, AND, NOT) и разделительные знаки (пробел, парные скобки). Как операнды в операциях сравнения используются имена Р-данных.

Пример: IF L=N THEN A := B ENDIF или IF L=N (A := B).

Здесь операторные скобки "THEN" и "ENDIF" и круглые парные скобки выполняют одну и ту же функцию. Внутри скобок может быть более одного оператора. В случае предыдущего примера оператор называется условным.

Если два или более ПРП выполняются при истинности одного и того же условия, то соответствующие операторы объединяются в предложение. Такое предложение является условным, а сочленение операторов называется **условным** сочленением.

Если условия двух или более условных предложений расположенных последовательно (не обязательно) взаимно исключают друг друга, то они также сочленяются в одно предложение. Такое предложение называется альтернативным предложением, а сочленение называется **альтернативным** сочленением. Прототип альтернативного сочленения – оператор CASE в традиционных алгоритмических языках.

Пример: Три условных предложения

```
IF A=1 (B := C).
IF A=2 (B := 2*C).
IF A=3 (B := 0).
```

Сочленены в одно альтернативное

```
CASE
        IF A=1 (B := C)
        IF A=2 (B := 2*C)
        IF A=3 (B := 0)
ENDCASE
```

Здесь CASE и ENDCASE операторные скобки, ограничивающие условное альтернативное предложение.

Если в условных или в альтернативных предложениях сочленяются только операторы, то эти предложения называются простыми условными. Простые условные предложения могут рассматриваться как неделимые целые и как компоненты (операторы) могут сочленяться в более сложные конструкции.

Предложение, составленное из сочленений хотя бы из двух вышеупомянутых видов и имеющее в качестве компоненты предложение, а не только операторы, называется сложным. Сложное предложение тоже можно рассматривать как неделимое целое и использовать его как компоненту. Таким образом, может порождаться сложная конструкция, составленная из операторов, имеющая иерархическую структуру.

Условные, альтернативные и сложные предложения тоже могут быть зависимыми или независимыми.

Реализация DM. Для записи процедур DM применяется несколько видов алгоритмических конструкций, в которых используются оператор чтения (READ) и оператор записи (WRITE). Прототип этих операторов – это операторы ввода-вывода в традиционных алгоритмических языках.

Оператор чтения – READ, использует указатель, сигнализирующий о том, закончилась или нет запись на А-ленте. Если при попытке чтения с А-ленты очередная запись прочитана и соответствующее А-данное доступно в А-памяти, то значение указателя “Y”, иначе “N”. Оператор ввода работает следующим образом.

Если при попытке чтения с А-ленты доступна очередная запись, то содержащееся там А-данное переносится в А-память. А-головка перемещается на одну запись вперед. Указатель устанавливается в состояние “Y”. Если на А-ленте очередная запись отсутствует, А-головка не смещается, а указатель устанавливается в состояние “N”. Имя указателя D_An, где n – идентификатор А-ленты.

Повторяющееся выполнение оператора ввода реализуется с помощью конструкции, которая в традиционных алгоритмических языках называется циклом.

В большинстве строчных алгоритмических языков заголовок и тело цикла текстуально неразрывны. Временный язык представляет оператор цикла двумя предложениями, связь между которыми устанавливается по идентификатору. Это сложное предложение. Это подобно тому как реализована связь между заголовком и телом цикла в ранних версиях КОБОЛа.

```
        PERFORM AA (условие)
        .
        .
AA:  BEGIN… END AA.
```

Где PERFORM – имя оператора цикла. AA – имя тела цикла, по которому реализуется связь между заголовком и телом цикла. Условие в скобках – это условие, при достижении которого прекращается выполнение цикла. Здесь может быть указанно также количество повторений. BEGIN и END – операторные скобки, ограничивающие тело цикла.

Далее пример размещения конкретного С-данного в области (рис. 4).

FS: Z = {B}, B – простое С-данное.

B = {$\cup$ C}, C – простое Эл-данное.

RG: R^{v} = < R2 [B]>,$R2^{\mu}$ =< R21[C] >.

А-лента имеет идентификатор A1. DM представляется операторами следующим образом.

```
        PERFORM AA ((D_A_A1 = “N”))
        .
        .
AA:     BEGIN
        READ A1
        END AA.
```

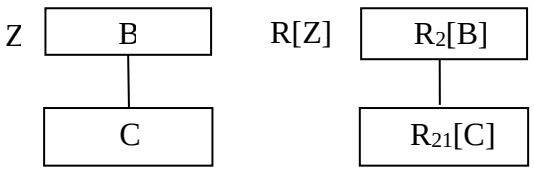


Рис. 4. Размещение расширенного А-данного Z в области R:
а – дерево типов свойств;
б – структура подобластей

При реализации процедур DM может иметь условие начала – S_{begin}. В тексте алгоритма условие реализуется условным и представляется оператором или предложением. При реализации DM могут иметь место также процедуры инициализации и (или) завершения.

После того, как простое Д-данное перемещено из записи в А-память, доступ к компонентам А-данного по именам. В операторе ввода указывается имя идентификатора А-ленты.

Реализация САЗ. САЗ[1] реализуется с помощью цикла. Частью этого цикла, точнее, частью тела, есть действие, которое сохраняет каждую реализацию Эл-данного С на А-ленте.

Пример: В – простое С-данное; С – простое Эл-данное; В = { ∪С }; Е – Эл-данное размещенное в А-памяти;

RG: R[B]; R^{μ} = < R_1[C] >;

R размещено на А-ленте, идентификатор которой А1. Между Е и В существует зависимость. Алгоритмическая конструкция состоит из двух частей.

PERFORM BB (“завершение вычислений”).

BB: BEGIN; (“реализация С = Е”); WRITE R_1; END BB.

Здесь “завершение вычислений” – это условие завершения вычислений. Условие должно быть задано при описании DM.

Здесь “реализация С = Е” – это оператор присваивания или предложение, которое выполняет формирование каждой из компоненты С, используя Е. Это может быть просто вычитание, сравнение и установление некоего признака или любое выражение, порождаемое прикладной областью и реализуемое языком. При одном выполнении тела цикла вычисляется одна компонента С.

САЗ[2] реализуется с помощью цикла, его неотъемлемая часть – это действия, обеспечивающие доступ к каждому Эл-данному или А-данному, которое является компонентой С-данного. Конструкция цикла приводится в следующем примере:

В – простое С-данное; С – простое Эл-данное; В = { ∪С }; D – Эл-данное размещенное в А-памяти;

RG: R[B]; R^{μ} = < R_1[C] >;

R размещено на А-ленте, идентификатор которой А1. Между D и В существует некая зависимость, реализующаяся предложением “реализация D = C”. Алгоритмическая конструкция состоит из двух частей.

READ A1
PERFORM BB (D_A_{A1} = “N”).

BB: BEGIN; (“реализация D = C”); READ A1; END BB.

Здесь “реализация D = C” – это оператор присваивания или предложение, выполняющее обработку или формирование D используя С. Предложение может содержать любое арифметическое или алгебраическое выражение, содержащее С как операнд. Любое выражение в смысле тех возможностей, которыми можно располагать в практической реализации DS-теория как средство описания схем декомпозиции и связанных с ними расчетов. Если это оператор присваивания, то он имеет такое выражение в правой части, после выполнения в цикла должен быть обеспечено формирование Эл-данного D.

Предварительное описание структуры программы

Компоненты тела фрейма. С каждым узлом DPS может быть связано произвольное количество А-зависимостей, которые могут быть реализованы и простыми предложениями, и сложными. Все А-зависимости, что соотносятся с одним

узлом DPS, ранее получили название АКУ. Все предложения алгоритмического языка, что представляют АКУ, называются параграфом. Совокупность всех параграфов – это программа. Она тоже может быть представлена деревом. С каждым узлом этого дерева связывается один параграф. Далее это дерево называется деревом алгоритма (DA). Как АКУ, так и параграф соотносятся с главным А-данным.

Как упоминалось раньше, с одним узлом DA может быть соотнесено DM и (или) более (не менее) одной САЗ[1]. Заголовки циклов, реализующих DM и САЗ1 заменяются одним заголовком, так как все они определяют одинаковый цикл. Все АК, что являются телом цикла для DM и каждой САЗ[1], объединяются в одно тело. Связь между параграфом с этим заголовком и параграфом с соответствующим объединенным телом цикла – это единственная связь между данными параграфами. Эта связь называется **иерархическим** сочленением. Оно используется для сочленения параграфов. Иерархическое сочленение между двумя смежными узлами может быть только одно. В изображении DA ребро, соединяющее два узла – родителя и потомка указывает на то, что параграфы, соответствующие этим узлам, будут сочленены иерархическим сочленением. Связь между всеми параграфами программы изображена DA. В этом дереве узлы соотносятся с параграфами программы.

Вся совокупность иерархически сочлененных параграфов называется программой. Программа содержит параграфы, количество которых соответствует количеству узлов DA. Параграфы сочленены иерархическим сочленением в таком же порядке, как соединены соответствующие узлы DPS. То есть, деревья DPS и DA – изоморфны.

Между синтаксическими средствами языка существует зависимость, показанная на рис. 5.

Между параграфами программы есть различия в зависимости от того, где расположены соответствующие им узлы DA.

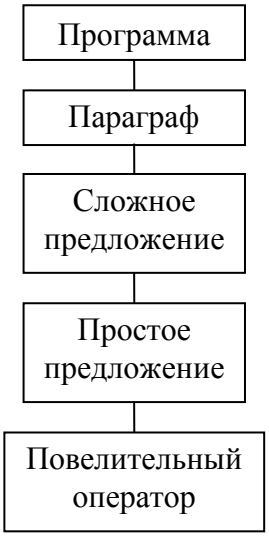


Рис. 5. Структура синтаксических конструкций изобразительного языка DS-теории

1. Корневой узел. Здесь могут быть предложения, реализующие заголовки циклов, САЗ и DM. Здесь не может быть тел циклов.

2. Концевой узел. Предложения, реализующие тела циклов САЗ и (или) DM узла-родителя. Здесь не может быть заголовков циклов.

3. Промежуточный узел. Здесь могут быть предложения – заголовки циклов, реализующие собственные САЗ и (или) DM и предложения тела циклов, реализующие САЗ и (или) DM узла-родителя.

ААЗ может быть реализовано в параграфах, соответствующих любому узлу DA.

Порядок выполнения операторов связанных с узлом DA зависит от нескольких факторов:

1) содержимое АКУ;

2) порядок размещения А-фрагментов на входной А-ленте;

3) порядок размещения А-фрагментов на выходной на А-ленте.

Представление АКУ

Далее перечислены ситуации размещения компонент главного А-данного.

Учитывая то, что внутри записи доступ к А-данным по именам, то порядок размещения Д-данных в записи не влияет на порядок размещения ПРП, обрабатывающие эти Д-данные.

Все предложения параграфа можно разбить на группы. Каждая из групп соотносится с одним А-фрагментом. Связь группы и А-фрагмента обусловлена тем, что в А-фрагменте размещены Д-данные, которые являются исходными для А-зависимостей реализованных предложениями группы. Далее эта группа будет называться АТ-предложением.

Порядок АТ-предложений в параграфе должен быть такой же, как порядок А-фрагментов на А-ленте.

А-фрагменты имеют порядок относительно границ области главного А-данного. Если все А-фрагменты главного А-данного имеют на А-ленте определенные места, то соответствующие им АТ-предложения должны быть записаны в том же порядке. АТ-предложения будут сочленены последовательным сочленением. Но возможна ситуация, когда АТ-предложения два или более зависимы и порядок их сочленения не соответствует порядку А-фрагментов на А-ленте. В этом случае параграф не может быть скомпонован. В практике программирования ищут возможность изменить порядок размещения А-фрагментов и разместить А-фрагменты соответственно порядку АТ-предложений. Также могут быть выполнены повторные проходы вдоль А-ленты.

А-фрагменты имеют относительный порядок следования. Ситуация, когда А-фрагменты имеют порядок следования по отношению друг к другу подобная размещению с порядком относительно границ области. Порядок АТ-предложений должен быть таким же, как и порядок соответствующих А-фрагментов. Сочленение АТ-предложений будет последовательным. Если же А-фрагменты будут разделятся непродуктивными А-фрагментами, то каждое АТ-предложение должно быть условным, а сочленение их должно быть альтернативным. Суть условий в том, чтобы распознать непродуктивный А-фрагмент и пропустить его на А-ленте.

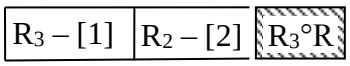
Возможна ситуация, когда АТ-предложения два или более зависимы и порядок их сочленения не соответствует порядку А-фрагментов на А-ленте, размещенных относительно друг друга. В этом случае параграф не может быть скомпонован. В практике программирования ищут возможность изменить порядок размещения А-фрагментов и разместить А-фрагменты соответственно порядку АТ-предложений. Также могут быть выполнены повторные проходы вдоль А-ленты.

Совместное размещение компонент первого вида (mix1). Все А-фрагменты, как компоненты главного А-данного, перемешаны по виду mix1. Соответствующие им АТ-предложения соединены альтернативным сочленением. Условия должны проверять (различать) встретившийся А-фрагмент и выполнять соответствующее АТ-предложение. Весь такой параграф – это одно альтернативное сочленение.

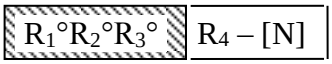
Совместное размещение компонент второго вида (mix2). При таком размещении вся область главного А-данного рассматривается разделенной на зоны. В зонах объединяются А-фрагменты с относительным порядком размещения. Эти зоны и остальные А-фрагменты рассматриваются как совокупность компонент на А-ленте, которые перемешаны по виду mix1. Для компонент с размещением mix1 используется альтернативное сочленение соответствующих АТ-предложений. Внутри зон где компоненты размещены относительно друг друга используется последовательное сочленение с уточнениями описанными выше для такого случая.

Совместное размещение компонент третьего вида (mix3). При таком размещении (рис. 6) А-фрагменты могут иметь определенные позиции в начале (а) или в конце (б) области занимаемой главным А-данным. Компоненты главного А-данного, у которых определенные позиции, могут быть также и в начале и в конце области (в).

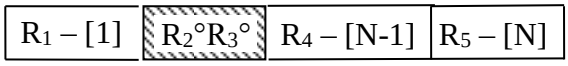
При таком размещении область рассматривается разделенной на зоны. Зоны А-фрагментов с определенными местами и зоны, где А-фрагменты перемешаны по виду mix1.



а



б



в

Рис. 6. Размещение А-фрагментов в области главного А-данного. Число в прямоугольных скобках рядом с именем А-фрагмента указывает на его позицию во включающей его подобласти.
N-1 – предпоследняя позиция,
N – последняя

АТ-предложения, соответствующие А-фрагментам, размещенным в зоне с определенными позициями, соединяются последовательным сочленением. Полученную АК временно (в условиях примера) назовем А. АТ-предложения, соответствующие А-фрагментам, размещенным в зоне с совместным размещением, соединяются альтернативным сочленением. Полученную АК назовем В. Получившиеся конструкции (А и В) соединяются последовательным сочленением в том порядке, который есть между соответствующими зонами. В условиях примеров на рис. 6. соединения будут следующими:

а: АК А + АК В
б: АК В + АК А
в: АК А + АК В + АК А

Любое из этих соединений – это параграф, соотносящийся с главным А-данным.

Условные А-фрагменты. Некоторые А-фрагменты к моменту расчета могут отсутствовать. Такая ситуация может быть обусловлена природой и значениями Р-свойств и, соответственно, Р-данных. Информация об их наличии может быть известна к началу расчета или нет. То, что запись или подобласть отсутствует, может стать известным только после того, как обработаны все компоненты главного А-данного или достигнут конец содержащей их подобласти. Существование таких компонент следует учитывать при сочленении АК в параграфе.

Условными могут быть А-фрагменты с определенными местами. Соответствующие АТ-предложения должны быть условными. В условии проверяется тип встретившийся записи или подобласти. Алгоритмическая конструкция будет выполнена, если встретится соответствующая компонента. В условии могут быть проверены Д-данные, к которым есть доступ и которые могут содержать признак наличия условного Д-данного.

Возможна ситуация, когда признаком конца подобласти есть А-фрагмент с определенным местом. Если эта компонента условная, то в качестве признака завершения подобласти необходимо учитывать также следующую безусловную компоненту.

Пример (рис. 7). Если условных А-фрагментов (записи A_2, A_3, … A_n) с определенными местами, которые являются условием завершения подобласти, более одной, то они все должны быть учтены в условии, проверяющем признак завершения подобласти. Условием завершения подобласти A_1, будет первая из встретившихся записей A_2, A_3, … A_n.

Подобласть A_1	A_2	A_3	…	A_n

Рис. 7. Обусловленные А-фрагменты с определенными местами как признаки завершения подобласти

Наличие условных компонент необходимо учитывать при сочленении АК порождаемых Р-данными, имеющими относительный порядок следования. Как и в случае Д-данных с определенными позициями, сочленяемые АК должны дополняться условием проверки их наличия. Для проверки используется либо А-данное со служебным Эл-данным признаком, либо наличие следующей подобласти. Как и в случае условных А-фрагментов с определенной позицией, если условные компо-

ненты с относительным порядком следования используются при определении конца подобласти, то условие завершения подобласти должно учитывать условные компоненты.

Обязательные А-фрагменты. При размещении компонент главного А-данного возможна ситуация, когда некоторые компоненты должны присутствовать обязательно. При совместном размещении должно быть определено служебное Эл-данное, выполняющее функцию флажка. Флажок должен быть установлен в состояние "компонента отсутствует" перед началом работы с компонентами, имеющими совместное размещение, а затем в процессе их обработки, когда обязательная компонента встретится, флажок переустанавливается в состояние "компонента присутствует". Если компонента не встретилась, то это рассматривается как ошибка спецификации на основе, которой строилась DS. С каждой обязательной компонентой соотносится собственный флажок.

Структура программы

Все предложения программы можно разделить на две категории. К первой категории относятся предложения, которые непосредственно реализуют обработку А-зависимостей. В этих предложениях используются имена Эл-данных (и, соответственно, Эл-свойств). Ко второй категории относятся предложения, реализующие САЗ обеих видов, DM, доступ к Д-данным на А-ленте, условные операторы, существование которых обусловлено размещением Д-данных в областях и записях на А-ленте. Совокупность всех предложений первой категории параграфа называется функциональным ядром (ФЯ) этого параграфа. Совокупность всех предложений второй категории параграфа называется алгоритмическим фреймом (АФ) этого параграфа.

Рассмотрим DS, DA имеющих два узла (рис. 8). В параграфе уровня K_0 должен быть заголовок цикла, реализующее собственную САЗ. В АТ-предложении, соответствующее уровню K_1, должен быть оператор, реализующий тело этой САЗ (уровня K_0). На уровне K_0 должен быть также заголовок цикла, обеспечивающий реализацию расчета ААЗ, соотносящиеся с уровнем K_1. Также должен быть заголовок цикла, обеспечивающий доступ к записям на А-ленте.

Все три вышеупомянутые циклы управляемы одним и тем же параметром цикла – количеством записей. Эти три цикла синтезируются в один. Заголовок общий для всех, а в теле цикла будут следующие компоненты:

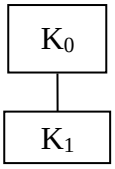


Рис. 8. Двухуровневая DA

АТ-предложение уровня K_1, включающее операторы, реализующие ААЗ уровня K_1, и операторы, реализующие тело цикла САЗ уровня K_0.

Порядок и характер сочленений операторов, реализующих А-зависимостей уровнях K_1 и K_0 зависят от размещения данных в областях и будет реализован соответствующими сочленениями в соответствующих АТ-предложениях. Количество записей неизвестно. Идентификатор А-ленты А1.

В параграфах уровня K_0 в первом и втором случае операторы BEGIN, READ A1, PERFORM, END AA – это алгоритмический фрейм, обеспечивающий реализацию САЗ уровня K_0. Алгоритмический фрейм обеспечивает реализацию функциональности уровня K_0. В параграфах уровня K_1 в обоих случаях операторы BEGIN, READ A1, END BB тоже алгоритмический фрейм, обеспечивающий реализацию функциональности уровня K_1.

Параграф, соответствующий уровню K_0

AA: BEGIN
READ A1
Оператор 1

PERFORM BB (D_A_{A1} = "N").

Оператор K

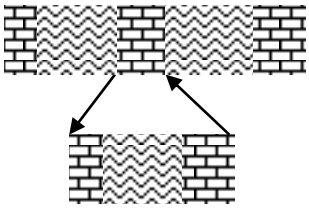
END AA

Параграф, соответствующий уровню K_1
BB: BEGIN
Оператор 1

Оператор M
READ A1;
END BB

Совокупность алгоритмических фреймов всех параграфов программы с учетом их сочленений и взаимного размещения называется алгоритмической матрицей (АМ) программы. Совокупность всех функциональных ядер всех параграфов программы с учетом их сочленений и взаимного размещения называется функциональным содержанием (ФС) программы.

Графически алгоритм как сочетание алгоритмической матрицы и функционального содержания показан на рис. 9. Две стрелки отражают передачу управления от заголовка к телу цикла. Но дальше иерархическое сочленение между параграфами будет изображаться одной линией (но не стрелками). Штриховка типа “кирпичная кладка” изображает компоненты алгоритмических фреймов или алгоритмической матрицы. Штриховка типа “волна” изображает компоненты функциональных ядер или функционального содержания алгоритма.



Параграф K_0.

Параграф K_1.

Примечание:

– компоненты алгоритмических фреймов;

– компоненты функциональных ядер.

Рис. 9. Графическое представление программы

Дополнительные замечания

Исходными данными для генерации алгоритма выступает DS с кортежами описаний узлов. С точки зрения более доступного изложения, DS можно рассматривать как схему алгоритма высокого уровня абстракции. Иными словами, традиционная блок-схема алгоритма или иное графическое изображение движения управления или потока данных, строится на основании DS. Хотя с точки зрения реального практического программирования в этом нет необходимости. Результат возможной генерации на настоящем уровне развития *DS*-теории – это текст программы на алгоритмическом языке. Этот текст не предназначен для практической работы программиста с ним. Текст – это промежуточный результат и рассматривается как исходные данные для последующей компиляции, и создания работающей прикладной программы, и нужен он только в период тестирования возможного генератора алгоритмов.

Естественное желание исследователей в сфере программирования – это желание формализовать процесс программирования. Программу можно рассматривать как результат некоторых операций над операндами. Если продолжить поиск в этом направлении, то появится понятие неких примитивов, операций над ними и конструкций составленных из примитивов. Такой подход позволяет применить или разработать математический аппарат для анализа, верификации и исследования программ. На этом пути появились идеи структурного программирования [4]. В рамках *DS*-теории идея “операция-операнд” имеет место и в отношении DS, и в отношении алгоритмов, записываемых с помощью алгоритмического языка. В статье описаны операции и операнды в отношении алгоритмов. Операции – это последовательные, условные, альтернативные и иерархические сочленения. Операнды – это АК и ПРП, реализующие DM и A-зависимости.

По мере того, как при реализации КА учитываются алгоритмически релевантные факторы другого типа, в DA появляются дополнительные узлы-параграфы

и вследствие этого изоморфизм деревьев DPS и DA разрушается. Это явление будет описано в следующей статье.

Выводы

В данной работе описаны способы размещения P-данных на A-ленте. Внесение в КА изменений, обусловленных способами размещения P-данных, позволяет создать текст программы, что приближает DS, как теоретическую модель, к реальным прикладным алгоритмам.

Анализ и исследование КА, представленного в виде текста на алгоритмическом языке, позволяет определить ряд понятий: функциональное ядро, алгоритмический фрейм, функциональное содержание и алгоритмическая матрица. Данные понятия необходимы для дальнейшего исследования и синтеза алгоритмов.

DS, как теоретическая модель ориентированная на разработку реальных прикладных программ, требует, используя специфические понятия, описывать некоторые факторы задачи или прикладной области на ранних этапах, начиная с исследования. Данными факторами являются КС, С-свойства и некоторые другие. Подобное описание задачи в рамках *DS*-теории создает предпосылки для постепенного упразднения этапов алгоритмизации и программирования, как наиболее насыщенных в интеллектуальном плане.

1. *Колесник В.Г.* DS-теория как прототип теории прикладных алгоритмов // Проблеми програмування. – 2012. – № 1. – С. 17–33.
2. *Robert Harper*. Existential Type. What, If Anything, Is A Declarative Language? 18 July 2013.
3. *Lloyd J.W*. Practical Advantages of Declarative Programming. GULP-PRODE'94 1994 Joint Conference on Declarative Programming. Peniscola (Spain), September 19–22, 1994.
4. *Дал У., Дейкстра Э., Хоор К*. Структурное программирование (Математическое обеспечение ЭВМ). – М.: Мир, 1975. – 248 с.

Получено 24.09.2014

Об авторе:

Колесник Валерий Георгиевич,
старший научный сотрудник
кафедры АПП.

Место работы автора:

Донбасская государственная
машиностроительная академия.
г. Краматорск,
ул. Шкадинова, 72, п/я 13.