УДК 004.421

**А.Е. ДОРОШЕНКО, В.Г. АКУЛОВСКИЙ**

# НИСХОДЯЩЕЕ ПРОЕКТИРОВАНИЕ АЛГОРИТМОВ В РАМКАХ АЛГЕБРОАЛГОРИТМИЧЕСКОГО ПОДХОДА

***Анотація.*** *Показана можливість декомпозиції алгоритмічних конструкцій, що дозволяє реалізувати стратегію низхідного проектування алгоритмів у межах алгеброалгоритмічного підходу.*
***Ключові слова:*** *алгоритми, декомпозиція алгоритмічних конструкцій, низхідна стратегія проектування алгоритмів, система алгоритмічних алгебр.*

***Аннотация.*** *Показана возможность декомпозиции алгоритмических конструкций, что позволяет реализовать стратегию нисходящего проектирования алгоритмов в рамках алгеброалгоритмического подхода.*
***Ключевые слова:*** *алгоритмы, декомпозиция алгоритмических конструкций, нисходящая стратегия проектирования алгоритмов, система алгоритмических алгебр.*

***Abstract.*** *Decomposing possibility of algorithmic constructions permitting to realize top-down design strategy of algorithms within the framework of algebra-algorithmic approach is shown.*
***Keywords:*** *algorithms, algorithmic constructions decomposing, top-down strategy of algorithms planning, system of algorithmic algebras.*

## 1. Введение

Развивая алгеброалгоритмический подход к программированию [1], с учетом важности роли, которую играют данные [2, 3], авторы предложили систему алгоритмических алгебр (САА/Д). САА/Д представляет собой двухосновную алгебраическую систему $<U, L, \Omega>$, основами которой являются множество Д-операторов $U$ и множество логических условий $L$, а $\Omega$ – её сигнатура, состоящая из $\Omega_1$-операций, принимающих значения на множестве $U$, и $\Omega_2$ – логических операций, принимающих значения на множестве $L$ [4, 5].

В рамках этого формального аппарата для описания алгоритмов используются Д-операторы, то есть операторы вида (D)X(D′), на входе и выходе которых специфицированы обрабатываемые данные. Использование Д-операторов обеспечивает возможность “имплантации” данных в формальный аппарат.

Отметим, что целью разработки подавляющего числа алгоритмов является его запись на некотором языке программирования, который назовем целевым. При этом требуется, чтобы полученные в результате разработки алгоритмические конструкции были достаточно просты и не вызывали затруднений при их реализации на этом языке программирования.

Данная задача решается в рамках нисходящей стратегии проектирования алгоритмов, и обеспечение возможности реализации такой стратегии в рамках алгебраического аппарата является целью данной работы.

Решение задачи заключается в том, чтобы в результате декомпозиции исходных алгоритмических конструкций получать более детализованные описания алгоритма. В данном случае в качестве исходных и производных алгоритмических конструкций рассматриваются Д-операторы.

Описание алгебры в объеме, необходимом для понимания полученных результатов, приведено в разд. 2.

© Дорошенко А.Е., Акуловский В.Г., 2012

ISSN 1028-9763. Математичні машини і системи, 2012, № 3

## 2. Алгебра алгоритмов с данными

САА/Д построена в результате модификации известной модели ЭВМ Глушкова. Модифицированная модель ЭВМ, представляющая собой (аналогично модели ЭВМ Глушкова) два взаимодействующих автомата – управляющий (У) и операционный (О) – дополнена внешней средой (ВС) и показана на рис. 1.

Внешняя среда (память, внешние устройства) – это множество носителей (источников и\или приемников) данных, которые определены следующим образом.

**Определение 1.** Данными называется упорядоченная пара $D=\langle A_j, K\rangle$, где $A_j$ – адрес носителя данных $A_j \subseteq A$, где $A$ – множество адресов, $K=\langle z_1,...,z_n\rangle$ – кортеж значений, хранимый носителем данных и изменяющийся в процессе выполнения алгоритма. Состояние данных определяется текущим кортежем значений.

Модель функционирует следующим образом.

Д-операторы являются управляющими воздействиями, последовательно подаваемыми автоматом У на вход автомата О. Последний выполняет каждый поступивший Д-оператор, в результате чего изменяются данные во внешней среде.

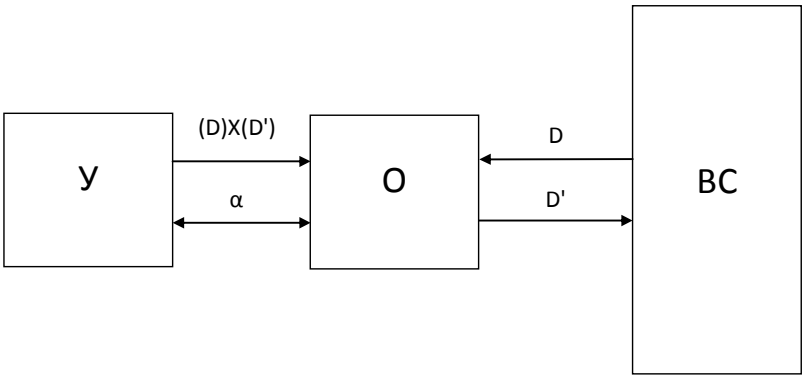

Рис. 1. Модифицированная модель ЭВМ

Логическое условие, характеризующее значения некоторого подмножества данных ВС, передаётся с выхода операционного автомата на вход управляющего, замыкая обратную связь между этими автоматами. Управляющий автомат, в соответствии со значением полученного логического условия, подает на вход автомата О один из нескольких возможных Д-операторов, который становится следующим в кортеже Д-операторов, поступающих с выхода автомата У.

Функционирование модели носит дискретный характер, а каждым шагом этого процесса является выполнение очередного Д-оператора. Между завершением выполнения предшествующего и началом выполнения следующего Д-оператора никакие действия моделью не выполняются. Такой пошаговый процесс функционирования модели продолжается до его завершения или может продолжаться неопределенно долго.

Для алгебраического аппарата, основанного на данной модели, принципиально важным является понятие «состояние вычислительного процесса». Это состояние $D^T=\{D_1,...,D_n\}$ определяется значениями, принятыми всеми данными ВС.

**Определение 2.** Д-оператор $(D)X(D')$ изменяет состояние вычислительного процесса (ВП), переводя его из любого исходного состояния $D_i^T$ в результирующее состояние $D_{i+1}^T$. Состояние ВП изменяется в результате обработки Д-оператором $(D)X(D')$ данных $D \subseteq D_i^T$, специфицированных на его входе, и продуцирования данных $D' \subseteq D_{i+1}^T$, специфицированных на его выходе. При этом для множеств данных $D'' = D \setminus D'$ и $D''' = D_{i+1}^T \setminus D'$ выполняются соотношения $D_i^T \supseteq D'' \subseteq D_{i+1}^T$ и $D_i^T \supseteq D''' \subseteq D_{i+1}^T$. То есть данные, оставшиеся неизменными после выполнения Д-оператора, имеют место как в предшествующем, так и в следующем за выполнением Д-оператора состоянии ВП.

В связи с определением 2 сделаем следующее небольшое отступление.

Из определения известно, что Д-оператор, обрабатывая специфицированные данные, переводит ВП из состояния $D_i^T$ в состояние $D_{i+1}^T$, то есть делает это за один шаг. При достаточной сложности обрабатываемых данных и алгоритма обработки в нарушение требования, сформулированного во введении, программная реализация такого Д-оператора вызовет существенные затруднения. То есть необходимо разбить переход из исходного состояния в результирующее на некоторую последовательность более простых шагов. Этим, собственно, и обусловлена необходимость декомпозиции Д-операторов.

Наконец приведем единственную из сигнатуры алгебры операцию, которую будем использовать в данном случае.

Композиция Д-операторов (операция обозначается "*") $(D_1)X_1(D_1')*(D_2)X_2(D_2')$ означает последовательное выполнение сначала Д-оператора $(D_1)X_1(D_1')$, затем Д-оператора $(D_2)X_2(D_2')$. То есть выполнение Д-оператора $(D_1)X_1(D_1')$ непосредственно предшествует выполнению Д-оператора $(D_2)X_2(D_2')$, а выполнение Д-оператора $(D_2)X_2(D_2')$ непосредственно следует за выполнением Д-оператора $(D_1)X_1(D_1')$.

По поводу композиции Д-операторов будем утверждать следующее.

**Утверждение.** В результате выполнения композиции Д-операторов $(D_1)X_1(D_1')*(D_2)X_2(D_2')$ ВП последовательно переходит из некоторого исходного состояния $D_i^T$ в состояния $D_{i+1}^T$ и $D_{i+2}^T$, для которых выполняется $D_1 \subseteq D_i^T$, $D_1', D_2 \subseteq D_{i+1}^T$ и $D_2' \subseteq D_{i+2}^T$.

Доказательство очевидно и следует из определения 2, определения операции и описания модели ЭВМ, из которой известно, что в промежутке между выполнением Д-операторов данные не изменяются.

На основе приведенных элементов САА/Д будем решать поставленную задачу.

### 3. Декомпозиция Д-операторов

Рассмотрим возможность декомпозиции Д-оператор $(D)X(D')$, то есть представление его в виде композиции двух других Д-операторов $(D)X(D')=(D_1)X_1(D_1')*(D_2)X_2(D_2')$, где Д-оператор $(D)X(D')$ называется исходным, а Д-операторы, связанные операцией «композиция $(D_1)X_1(D_1')*(D_2)X_2(D_2')$», – производными.

Здесь уместно привести следующую цитату.

"Стало ясно, что решения о структурировании данных нельзя принимать без знания алгоритмов, применяемых к этим данным, и наоборот, структура и выбор алгоритмов существенным образом зависят от структуры данных. Говоря короче, строение программ и структуры данных неразрывно связаны" [6, стр. 8].

Из цитаты следует, что при декомпозиции Д-операторов совершенно необходимо учитывать обрабатываемые данные. В связи с этим рассмотрим особенности их обработки.

Во-первых, обработка данных, начатая Д-оператором, расположенным в композиции первым, может быть продолжена вторым Д-оператором. В связи с этой особенностью обработки данных введем следующее определение.

**Определение 3.** Д-операторы в композиции $(D_1)X_1(D_1')*(D_2)X_2(D_2')$ назовем информационно связанными (в дальнейшем связанными), если для специфицированных у них данных выполняется соотношение $D_1' \cap D_2 \neq \varnothing$, и несвязанными, если $D_1' \cap D_2 = \varnothing$. Данные $D^{св} = D_1' \cap D_2$ будем называть связывающими.

Во-вторых, каждый из Д-операторов, входящих в композицию, в общем случае обрабатывает и продуцирует некоторое подмножество данных, на которые другой Д-

оператор не влияет. В частном случае производные Д-операторы могут обрабатывать различные данные.

В связи со второй особенностью обработки данных введем следующее определение.

**Определение 4.** В композиции Д-операторов $(D_1)X_1(D_1')*(D_2)X_2(D_2')$ множества данных $D_1'^{СБ}=D_1'\setminus D^{СВ}$, $D_2^{СБ}=D_2\setminus D^{СВ}$ будем называть собственными. Для данных $D_2^{СБ}$, которые Д-оператором $(D_1)X_1(D_1')$ не обрабатываются, выполняется соотношение $D_2^{СБ}\cap D_1'=\varnothing$. Для данных $D_1'^{СБ}$, которые не обрабатываются Д-оператором $(D_2)X_2(D_2')$, – $D_1'^{СБ}\cap D_2=\varnothing$. Множества данных $D_2^{СБ}$ и $D_1'^{СБ}$ могут быть пустыми.

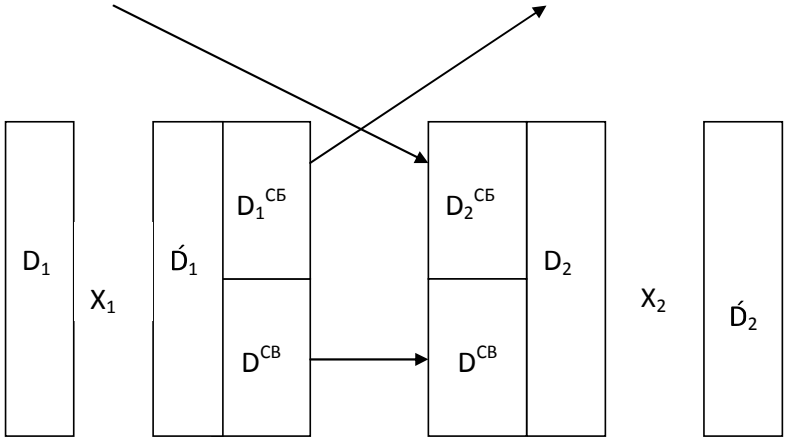


Рис. 2. Связывающие и собственные данные

Проиллюстрируем данные, введенные в определениях 3 и 4, с помощью рис. 2.

Принципиально важным для декомпозиции Д-операторов является то, что результат выполнения производных Д-операторов должен полностью совпадать с результатом выполнения исходного. В связи с этим введем понятие их эквивалентности.

**Определение 5.** Д-оператор $(D)X(D')$, переводящий ВП из состояния $D_i^T$ в состояние $D_{i+1}^T$, эквивалентен композиции Д-операторов $(D_1)X_1(D_1')*(D_2)X_2(D_2')$, переводящей ВП из состояния $D_j^T$ в состояние $D_{j+2}^T$, если при равенстве исходных состояний ВП $D_i^T=D_j^T$ в результате их выполнения будут получены результирующие состояния $D_{i+1}^T=D_{j+2}^T$.

Прежде чем приступить к доказательству реализуемости декомпозиции Д-операторов, остановимся на понятии элементарного Д-оператора.

Поскольку для общего случая сложно оценить трудность программной реализации Д-оператора, будем полагать, что Д-оператор, реализующий одну операцию целевого языка программирования, элементарный. Этот выбор весьма просто обосновывается: такой Д-оператор не нуждается в декомпозиции. Кроме того, будем полагать, что элементарный Д-оператор выполняется за один элементарный шаг ВП.

**Теорема.** Произвольный Д-оператор $(D)X(D')$, если он не элементарный, может быть декомпозирован, то есть представлен в виде эквивалентной ему композиции двух других Д-операторов $(D_1)X_1(D_1')*(D_2)X_2(D_2')$.

Доказательство.

Если Д-оператор $(D)X(D')$ не элементарный, то он реализует, по крайней мере, две операции целевого языка программирования, каждая из которых может быть описана с помощью Д-оператора. Таким образом, этот Д-оператор может быть декомпозирован.

В общем случае будем полагать, что исходный Д-оператор, переводящий вычислительный процесс из состояния $D_i^T$, такого, что $D\subseteq D_i^T$, в состояние $D_{i+1}^T$, такое, что $D'\subseteq D_{i+1}^T$, выполняет этот переход за несколько шагов в общем случае не элементарных. На каждом шаге выполняется некоторая последовательность операций, а в результате выполнения

указанной последовательности шагов данные $D$, специфицированные на входе Д-оператора, преобразуются в данные $D'$, специфицированные на его выходе.

Предположим, что после некоторого $i$-го шага обработки получены данные, подлежащие дальнейшей обработке $D^o$, и результирующие данные, которые дальнейшей обработке не подлежат, $D^r \subseteq D'$. А на следующем $i+1$-ом шаге продолжается обработка данных $D^o$ и начинается обработка множества данных $D^s \subseteq D$, которые до этого момента не обрабатывались.

Разобьем последовательность операций на два этапа таких, что первый завершается $i$-ым шагом, а второй начинается с $i+1$-го. Будем полагать, что каждый из этих этапов выполняется одним из производных Д-операторов.

В результате выполненного разбиения получаем композицию Д-операторов $(D_1)X_1(D_1')*(D_2)X_2(D_2')$, для которых выполняется $D_1 \subseteq D$ и $D_2' \subseteq D'$, а $D = D_1 \cup D^s$ и $D' = D_2' \cup D^r$. При этом данные множества $D^r$, в соответствии с определением 4, становятся собственными результирующими данными Д-оператора $(D_1)X_1(D_1')$, то есть $D^r = D'^{СБ} \subseteq D_1'$. А множество данных $D^s$ – собственными входными данными Д-оператора $(D_2)X_2(D_2')$, то есть $D^s = D^{СБ} \subseteq D_2$. Данные множества $D^o$, в соответствии с определением 3, являются связывающими, то есть $D^o = D^{СВ}$.

Из утверждения следует, что композиция Д-операторов переводит вычислительный процесс из состояния $D_j^T$ последовательно в состояния $D_{j+1}^T$ и $D_{j+2}^T$, где $D_1'^{СБ}, D_2^{СБ} \subseteq D_{j+1}^T$. Однако в связи с тем фактом, что данные $D_2^{СБ}$ остались неизменными в состоянии $D_{j+1}^T$, а данные $D_1'^{СБ}$ остаются неизменными при переходе ВП в состояние $D_{j+2}^T$, в соответствии с определением 2, $D_2^{СБ} \subseteq D_j^T$, а $D_1'^{СБ} \subseteq D_{j+2}^T$.

Таким образом, $D_i^T \subseteq (D_1 \cup D^s) = (D_1 \cup D_2^{СБ}) \subseteq D_j^T$, то есть $D_i^T = D_j^T$. А с учетом того, что последовательность операций, выполняемых над данными, в исходном и производных Д-операторах совпадает, то $D_{i+1}^T \subseteq D_2' \cup D^r = D_2' \cup D_1'^{СБ} \subseteq D_{j+2}^T$, то есть $D_{i+1}^T = D_{j+2}^T$.

Отсюда, в соответствии с определением 5, следует, что полученные производные Д-операторы эквивалентны исходному, а декомпозиция Д-операторов реализуема.

Теорема доказана.

**Следствие.** Для обеспечения эквивалентности исходного и производных Д-операторов на входе и выходе последних необходимо специфицировать собственные и связывающие данные. Если собственные данные у производных Д-операторов, в соответствии с определением 4, отсутствуют ($D_1'^{СБ} = \varnothing$, $D_2^{СБ} = \varnothing$), то $D_1 = D$, $D_2' = D'$.

## 4. Заключение

В приведенной теореме доказана возможность декомпозиции Д-операторов. В результате последовательной декомпозиции всех неэлементарных Д-операторов реализуется нисходящая стратегия проектирования алгоритмов. Поставленная в работе задача, таким образом, решена.

Процесс проектирования – творческий процесс, в существенной мере зависящий от класса решаемой задачи, о трудностях формализации которого говорил ещё Дейкстра. На текущий момент задача формализации этого процесса не решена и решение этой задачи, по крайней мере, частичное, является перспективным направлением дальнейших исследований.

## СПИСОК ЛИТЕРАТУРЫ

1. Алгеброалгоритмические модели и методы параллельного программирования / Ф.И. Андон, А.Е. Дорошенко, Г.Е. Цейтлин, Е.А. Яценко. – Киев: Академпериодика, 2007. – 634 с.
2. Шнейдерман Б. Психология программирования: человеческие факторы в вычислительных и информационных системах / Шнейдерман Б. – М.: Радио и связь, 1984. – 304 с.
3. Bastani F.B. The effect of data structures on the logical complexity of programs / F.B. Bastani, S.S. Iyengar // CACM. – 1987. – Vol. 30, N 3. – P. 250 – 259.
4. Акуловский В.Г. Основы алгебры алгоритмов, базирующейся на данных / В.Г. Акуловский // Проблеми програмування. – 2010. – № 2, 3. – С. 89 – 96.
5. Дорошенко А.Е. Алгебра алгоритмов с данными и прогнозирование вычислительного процесса / А.Е. Дорошенко, В.Г. Акуловский // Проблеми програмування. – 2011. – № 3. – С. 3 – 10.
6. Вирт Н. Алгоритмы + структуры данных = программы / Вирт Н. – М.: Мир, 1985. – 656 с.

*Стаття надійшла до редакції 07.06.2012*