УДК 004.42

АЛГЕБРА ДЛЯ ОПИСАНИЯ ДАННЫХ В КОМПОЗИЦИОННЫХ СХЕМАХ АЛГОРИТМОВ

В.Г. Акуловский

Академия таможенной службы Украины.
49000, г. Днепропетровск, ул. Дзержинского 2/4.
Тел/Факс: (0562) 745 5596, (0562) 47 1791
e-mail: academy@amsu.dp.ua.

Предлагается алгебра данных, в рамках которой реализуется комплексный подход к программированию при описании управляющих структур и структур данных в композиционных схемах алгоритмов.

The algebra of the data in which frameworks the complex approach to programming is realized at the description of operating structures and structures of the data in composite schemes of algorithms is offered.

Введение

В соответствии с важностью роли, играемой данными в программировании [1, 2], автором в работах [3, 4], заложены основы алгебраического аппарата, в котором, в результате модификации модели ЭВМ Глушкова [5], в описание управляющих структур алгоритмов органично "встроены" данные. Упомянутый формальный аппарат – это система алгоритмических алгебр (САА/Д) $<U; L; S>$, где U – множество Д-операторов, L – множество логических условий, S – сигнатура операций, состоящая из операций S_1, принимающих значения на множестве операторов U и логических операций S_2, принимающих значения на множестве L.

В рамках САА/Д в качестве одной из форм описания алгоритмов используются композиционные схемы (КС) [3, 4], которые записываются, в виде

$$(D)O(D') = (D_1)O_1(D'_1) * (D_2)O_2(D'_2) * ... * (D_k)O_k(D'_k), \quad (1)$$

где $(D)O(D')$ – исходный, а $(D_1)O_1(D'_1)$, $(D_2)O_2(D'_2)$, …, $(D_k)O_k(D'_k)$ – производные Д-операторы. То есть, исходный Д-оператор $(D)O(D')$ декомпозируется, а каждый производный Д-оператор $(D_i)O_i(D'_i)$, реализует некоторую часть функций исходного Д-оператора. При этом на входах Д-операторов специфицированы обрабатываемые, а на выходах результирующие данные, чем и обусловлено использование названия Д-оператор.

В КС, помимо данных специфицированных на входе и выходе Д-оператора $(D)O(D')$, имеют место промежуточные данные D^{pr}, которые, играя вспомогательную роль, служат для преобразования исходных данных в результирующие, а на входе и выходе Д-оператора $(D)O(D')$ не специфицируются.

Для данных D и D', специфицированных на входе и выходе исходного Д-оператора (D)O(D'), выполняются следующие соотношения:

$$D \subseteq D_1 \cup D_2 \cup ... \cup D_k, \quad (2)$$

$$D' \subseteq D'_1 \cup D'_2 \cup ... \cup D'_k, \quad (3)$$

где для любого D_i и D'_i может выполняться $D_i \cap D = \varnothing$ и $D'_i \cap D' = \varnothing$;

$$D^{pr} = (D_1 \cup D_2 \cup ... \cup D_k)/D) \cup (D'_1 \cup D'_2 \cup ... \cup D'_k)/D'. \quad (4)$$

То есть, $(D_i)O_i(D'_i)$ обрабатывает все или некоторую часть данных, специфицированных на входе исходного Д-оператора, при $D_i \cap D \neq \varnothing$ и продуцирует все или некоторую часть данных, специфицированные на его выходе, при $D'_i \cap D' \neq \varnothing$. В случае, когда $D_i \cap D = \varnothing$ и/или $D'_i \cap D' \neq \varnothing$, он обрабатывает и/или продуцирует промежуточные данные $D_i^{pr} \in D^{pr}$ и $D'^{pr}_i \in D^{pr}$.

Множества данных $D'' \subseteq D$ и $D''' \subseteq D'$ таких, что

$$D'' \subseteq D_i, \quad D'' \cap D_j \neq \varnothing \text{ и } D''' \subseteq D'_i, \quad D''' \cap D'_j \neq \varnothing \quad (5)$$

для всех i и j, целиком обрабатываются Д-оператором $(D_i)O_i(D'_i)$. То есть, с точки зрения описания этих данных в КС они на данном этапе разработки являются неделимыми. В противном случае данные будем называть делимыми.

Разработка управляющих структур алгоритмов неразрывно связана с проектированием структур данных. В работе [6] отмечалась целесообразность применения средств, используемых для описания управляющих структур, для описания структур данных. Такой формальный аппарат (алгебра структур данных, базирующаяся на теории модифицированной системы алгоритмических алгебр (САА-М)), ориентированный на порождение и распознавание данных, был предложен в работах [7, 8].

Однако, САА/Д имеет принципиальные отличия от САА-М, которые в первую очередь состоят в том, что данные наряду с описанием управляющих структур, являются неотъемлемой частью формализованного описания алгоритма. Это отличие является предпосылкой для пересмотра основополагающих подходов к программированию, применительно к некоторому классу программных систем.

Остановимся на упомянутых подходах.

При разработке алгоритмов описываются две их составляющие – управляющие структуры и структуры данных. В случае, когда первая составляющая более приоритетна, реализуется программирование от управления. В противном случае – программирование от данных. При этом не один из этих подходов, воплощениями которых являются методологии структурного и объектно-ориентированного программирования, не универсален. Это обуславливается тем, что существуют программные системы, которым не адекватен любой, в отдельности используемый, подход (соответственно, методология). Типичным представителем такого класса систем являются информационно-управляющие системы.

Исходя из этого, сформулируем цель данной работы.

Целью данной работы является построение алгебры данных, которая при разработке алгоритмов для вышеупомянутого класса систем обеспечит реализацию комплексного подхода к программированию, адекватного специфике этого класса. Комплексность подхода заключается в возможности использования любого из указанных подходов на различных этапах разработки одного алгоритма и совместного их использования в случае необходимости.

Кроме того, формальный аппарат должен обеспечивать согласованность при описании управляющих структур и структур данных, реконфигурацию данных, в случае изменения (модификации) алгоритма, а так же возможности контроля корректности описания и обработки данных.

Для достижения указанных целей необходимо определить формы записи структур данных, их свойства, способы доступа к ним и средства преобразования. Под доступом к данным будем понимать возможность спецификации требуемых данных на входе или выходе производного Д-оператора в КС.

Построение алгебраического аппарата начнем с формализации данных.

Формализация данных

Для формализации данных введем понятие носителя данных.

Атомарным носителем данных является бит (разряд). Заметим, что не обязательно бит. В качестве атомарного носителя данных может выступать, например, трит в случае трехзначного компьютера.

Будем полагать, что в нашем распоряжении имеется множество атомарных носителей данных $B=\{b_1,...,b_n\}$. Непрерывные последовательности атомарных носителей данных образуют элементарные (неделимые) носители данных $n_i=<b_m,b_{m+1},...,b_{m+s}>$ (в частном случае $n_c=b_p$). Размер (количество атомарных носителей) элементарного носителя данных определяет его информационную ёмкость. В общем случае существуют как различные $|n_i|\neq|n_j|$, так и одинаковые $|n_i|=|n_j|$ ($i\neq j$) по размеру носители данных.

Памятью данных будем называть перестановку элементарных носителей данных $N=\langle n_{a_1},n_{a_2},...n_{a_m}\rangle$ такую, что $n_{a_1}\cup n_{a_2}\cup...\cup n_{a_m}=B$, а для элементов индексного множества $A=\langle a_1,a_2,...,a_m\rangle$ выполняется $a_i<a_j$ при $i<j$.

Носители данных в общем случае определим следующим образом.

Определение 1. Носителями данных будем называть классы разбиения памяти $\boldsymbol{N}=\langle N_{A_i}|A_i\in A\rangle$, образующие семейство непустых и различимых подмножеств таких, что $N_{A_i}=\langle n_{a_i},n_{a_{i+1}},...n_{a_{i+k}}\rangle$, $A_i=a_i$, $\bigcup_{A_i\in A}N_{A_i}=\boldsymbol{N}$, $N_{A_i}\cap N_{A_j}=\varnothing$ для всех $A_i,A_j\in A$. Частными случаями носителей данных являются элементарные носители данных, когда $N_{A_i}=n_{a_i}$. Элементы множества $A_{\boldsymbol{N}}=\langle A_{i_1},A_{i_2},...A_{i_p}\rangle$ ($A_{\boldsymbol{N}}\subseteq A$), поставленные в однозначное соответствие носителям данных $N_{A_i}\in\boldsymbol{N}$ ($\boldsymbol{N}\sim A_N$), будем называть адресами носителей данных. Элементы множества $A_N=\langle a_i,a_{i+1},...,a_{i+k}\rangle$ ($A_N\subseteq A$), поставленные в однозначное соответствие элементарным носителям данных $n_{a_j}\in N_{A_i}$ ($N_{A_i}\sim A_N$) – адресами элементарных носителей данных.

Очевидно, что при различных разбиениях могут быть получены различные сочетания носителей данных, при этом существуют как различные $|N_{A_i}|\neq|N_{A_j}|$, так и одинаковые $|N_{A_i}|=|N_{A_j}|$ по размеру носители данных.

После определения носителей данных перейдем к рассмотрению понятия данные.

Элементарный носитель данных содержит (хранит) в каждый момент времени некоторое значение, записанное в некоторой форме (например, целое число или число с плавающей точкой), которое изменяется в ходе вычислительного процесса.

Информационная емкость элементарных носителей данных и формы хранимых значений определяют форматы данных, образующих множество $F=\{f_1,f_2,...f_k\}$. Множество значений $З=\{з_1^{f_j},...,з_p^{f_j}\}$, которые может хранить элементарный носитель данных, ограничены форматом этих значений $f_j \in F$. То есть значения, хранимые носителями данных, изменяются в диапазоне, определяемом форматом хранимых данных.

Отметим, что в частном случае в качестве значений, хранимых элементарным носителем данных, могут выступать адреса носителей данных.

Исходя из изложенного и определения 1, обозначив формат данных для общего случая f_x, определим их следующим образом.

Определение 2. Данными будем называть упорядоченную пару $D^{f_x}=\langle A_i,\langle K\rangle\rangle$, где $K=\langle з_1^{f_x},...з_n^{f_x}\rangle$ – кортеж значений, носителем которого является элемент разбиения N_{A_i}, расположенный по адресу $A_i \in A_N$. Кортеж K образован текущими значениями, хранимыми всеми элементарными носителями данных, входящими в N_{A_i}. Формат данных D^{f_x} определяется форматом значений, образующих кортеж K. Частным случаем данных являются простые данные $d^{f_x}=\langle A_i,з^{f_x}\rangle$. Простые данные $d^{f_a}=\langle A_i,з^{f_a}\rangle$ такие, что $з^{f_a}=a_j \in A$, то есть содержат адрес некоторых данных, расположенных по адресу a_j, называются ссылкой.

Будем говорить, что данные D^{f_x} расположены (размещаются) по адресу A_i и определим, связанное с размещением данных по адресам, следующее их свойство.

Определение 3. Данные $D_i^{f_x}$ и $D_j^{f_x}$ расположены последовательно, то есть $D_i^{f_x}$ непосредственно предшествует $D_j^{f_x}$, а $D_j^{f_x}$ непосредственно следует за $D_i^{f_x}$, если для адресов этих данных A_i и A_j выполняется $A_i < A_j$, и не существует данных $D_k^{f_x}$ с адресом A_k, для которого выполняется $A_i < A_k < A_j$. В противном случае будем говорить, что данные $D_i^{f_x}$ и $D_j^{f_x}$ расположены непоследовательно.

Определив данные, перейдем к рассмотрению регулярно организованных данных, обозначая совпадающие форматы данных f_y. Для этого, учитывая, что, в соответствии с определениями 1 – 3, $D^{f_x}=\langle d_1^{f_y},..., d_n^{f_y}\rangle$ – n-размещение простых данных, где $n=|K|$, а каждому элементу кортежа K в однозначное соответствие поставлен адрес, будем осуществлять разбиения множества D^{f_x}.

Первое разбиение ${}_1D_{j_1}^{f_x}=\langle D_{j_1}^{f_x} | j_1 \in J_1\rangle$ образует семейство непустых и различимых подмножеств таких, что $\bigcup_{j_1 \in J_1} D_{j_1}^{f_x}=D^{f_x}$ $D_{j_1}^{f_x} \cap D_{i_1}^{f_x}=\varnothing$ (при $j_1 \neq i_1$), $|D_{j_1}^{f_x}|=|D_{i_1}^{f_x}|$ (для всех j_1 и i_1), где $J_1=\langle 1,2,...,p_1\rangle$, $n_1 \times |D_{j_1}^{f_x}|=n$ и каждому $D_{j_1}^{f_x}$ в однозначное соответствие поставлен адрес $A_{j_i} \in A$. Аналогичное разбиение ${}_2D^{f_x}$ при тех же условиях выполним для каждого полученного класса $D_{j_1}^{f_x}$ предыдущего разбиения и т.д. Этот процесс продолжим до получения разбиения ${}_nD^{f_x}=\langle d_{j_1,j_2,...j_k}^{f_y} | j_k \in J_k\rangle$ такого, что $\bigcup_{j_k \in J_k} d_{j_1,j_2,...j_k}^{f_y}=D_{j_1,j_2,...j_{k-1}}^{f_x}$, где $J_k=\langle 1,2,...,p_k\rangle$.

В результате проведенных построений получены регулярно организованные данные, представляющие собой иерархию из *k* уровней, на каждом *i*-м уровне которой адресам, изменяются с постоянным шагом, в соответствие поставлен индекс с единичным шагом изменения.

Исходя из этого, массивы определим следующим образом.

Определение 4. Регулярно организованные данные D^{f_x}, элементы кортежа значений которых $K=\langle з_1^{f_y},...з_n^{f_y}\rangle$ имеют один формат, будем называть многомерным массивом простых данных и обозначать ${}^{n_1,...,n_k}D^{f_x}$, где $n_1,...,n_k$ – индексы, задающие размерности массива, для которых выполняется $n_1 \times ... \times n_k=|K|$.

Частным случаем многомерных массивов является вектор (одномерный массив), который запишем в виде ${}^nD^{f_x}$ ($n=|K|$), полученный в результате того, что адресу каждого элемента $d_j^{f_y} \in D^{f_x}$ в однозначное соответствие ставится значение индекса *j*.

Из соображений общности изложения, простые данные в случае необходимости будем трактовать как вырожденный случай массива и обозначать ${}^1D^{f_x}$.

Исходя из изложенного, введем следующее утверждение, предварительно оговорив, что в нем индексы массивов r_1, r_2, p_x будем трактовать как любые совокупности индексов, а последовательно расположенные данные будем записывать, используя в качестве разделителя символ “;”.

Утверждение 1. Любой массив, за исключением ${}^1D^{f_x}$, может быть преобразован к виду ${}^{r_1}D^{f_x}={}^{p_1}D_1^{f_x};...;{}^{p_s}D_k^{f_x}$, где ${}^{p_1}D_1^{f_x};...;{}^{p_s}D_k^{f_x}$ – произвольная совокупность последовательно расположенных массивов любой размерности, при условии $|{}^{p_1}D_1^{f_x}|+...+|{}^{p_s}D_k^{f_x}|=|{}^{r_1}D^{f_x}|$. Любая совокупность последовательно расположенных массивов,

образованных элементами одного формата, может быть преобразована к виду ${}^{p_1}D_1^{f_x};...;{}^{p_s}D_m^{f_x}={}^{r_2}D^{f_x}$, где ${}^{r_2}D^{f_x}$ – массив произвольной размерности, при условии $\left|{}^{r_2}D^{f_x}\right|=\left|{}^{p_1}D_1^{f_x}\right|+...+\left|{}^{p_s}D_m^{f_x}\right|$.

Доказательство. В соответствии с определением 4, ${}^{r_1}D^{f_x}=D^{f_x}$, а D^{f_x} может быть представлено в виде $D^{f_x}=D_1^{f_x}\cup...\cup D_k^{f_x}$, где все подмножества расположены последовательно, для всех j и i выполняется $D_j^{f_x}\cap D_i^{f_x}=\varnothing$ и $\left|D^{f_x}\right|=\left|D_1^{f_x}\right|+...+\left|D_k^{f_x}\right|$. В результате выполнения описаного выше процесса разбиения над всеми подмножествами $D_i^{f_x}$ будет получена совокупность массивов требуемой размерности, в частности векторов. Если $D_j^{f_x}$ простые данные (${}^1D_j^{f_x}$), они сохраняется неизменными. Таким образом, требуемое соотношение при указанном ограничении реализуемо.

Так как каждый массив ${}^{p_i}D_i^{f_x}$ получен в результате разбиения, то ${}^{p_i}D_i^{f_x}=D_i^{f_x}$. А так как, по условию утверждения, массивы ${}^{p_1}D_1^{f_x};...;{}^{p_s}D_m^{f_x}$ расположены последовательно и образованы элементами одного формата и для всех j и i выполняется $D_j^{f_x}\cap D_i^{f_x}=\varnothing$, то в результате объединения подмножеств получим данные $D_1^{f_x}\cup...\cup D_m^{f_x}=D^{f_x}$ такие, что $\left|D_1^{f_x}\cup...\cup D_m^{f_x}\right|=\left|D^{f_x}\right|$. Выполняя рассмотренное выше разбиение множества D^{f_x} получим многомерный массив ${}^{r_2}D^{f_x}$ требуемой размерности при указанном ограничении.

Утверждение доказано.

Следствие 1. Из доказательства утверждения легко увидеть, что любой массив ${}^{r_j}D^{f_x}$ размерности r_j может быть преобразован в массив ${}^{r_i}D^{f_x}$ размерности r_i, при условии $\left|{}^{r_j}D^{f_x}\right|=\left|{}^{r_i}D^{f_x}\right|$.

Теперь остановимся на таком важном аспекте описания данных, как их именование.

Очевидно, что данные однозначно идентифицируются своими адресами в памяти. При этом подчеркнем, что адресом любой структуры данных является адрес её первого компонента. Однако, для удобства описания алгоритмов, всем данным необходимо поставить в однозначное соответствие уникальные имена – идентификаторы.

Заметим, что, очевидно, следует придерживаться некоторых правил формирования идентификаторов. В частности, рационально использовать правила, характерные для целевого языка программирования (языка, на котором будет реализовываться разрабатываемый алгоритм).

В данном случае, из соображений общности, будем полагать, что мы располагаем некоторым конечным множеством произвольных идентификаторов I таким, что $\left|I\right|$ достаточна для описания любого алгоритма, любой идентификатор $X_i\in I$ уникален и ставится в однозначное соответствие адресу A_j данных D^{f_x}. Такой идентификатор замещает данные при описании алгоритма и обеспечивая доступ к данным расположенным по этому адресу.

Данные, снабженные идентификатором, будем называть поименованными. Поименованные простые данные будем называть переменной, а поименованную ссылку – указателем.

Именование массивов выполним традиционно.

Если многомерному массиву ${}^{n_1,...,n_k}D^{f_x}$ приписывается идентификатор M, то массив запишем в виде $M[n_1,...,n_k]$, элемент массива – $M[i_1,...,i_k]$, а идентификатор подмассива (сечения) формируются в результате отбрасывания размерностей справа налево, например, $M[n_1,...,n_{k-1}]$. Векторы именуются аналогично $V[n]$, $V[j]$, а подмассив $V[i,j]$, где индексы, определяющие границы подмассива, такие, что $0\le i<j\le(n-1)$.

Заметим, что индексы могут изменяться как от нуля, так и от единицы.

В результате формализации данных введены и поименованы простые данные, в частности указатели, и массивы произвольной размерности. В дальнейшем изложении введенные данные будем называть базовыми структурами, понимая под этим термином то, что они формируются в результате некоторой интерпретации структуры носителя данных.

На основе выполненной формализации, исходя из поставленной задачи, перейдем к построению алгебраического аппарата.

Алгебра данных

Учитывая пассивную роль, которую данные играют в алгоритмах и программах [9], в сигнатуру создаваемой алгебры включим как операции, имеющие аналоги в алгебре алгоритмов, так и операции таких аналогов не имеющие.

Операцию композиция определим с элементами рекурсии.

Операция композиция (*), определенная на множестве последовательно расположенных поименованных базовых структур данных и записей, $X_1*X_2=Y$ (в общем случае $X_1*...*X_k=Y$) описывает новые данные, называемые записью. Доступ к *i*-му элементу записи осуществляется посредством пары идентификаторов Y_X_i, разделенных символом "_". В случае вложенности записей – последовательностью

идентификаторов, разделенных символом "_", длина которой определяется глубиной вложенности. Если элементом записи является массив, доступ осуществляется с помощью идентификатора вида $Y\_X_i[j_1,...,j_k]$.

Исходя из определения операции, приведем следующее свойства записи Y:

$$Y * X = Y_1,\ X * Y = Y_1,\ X_1 * Y * X_2 = Y_1, \tag{6}$$

где X, X_1, X_2 – записи или базовые структуры данных, Y_1 – новая запись;

$$Y = X_1;...;X_n, \tag{7}$$

где X_i – запись или базовая структура данных.

На множестве последовательно расположенных поименованных базовых структур данных и записей определим операцию **р-итерация** $[n_1,...,n_k]\{X_1;...;X_k\}$. Эта операция, выполнив над этими структурами операцию композиция, и, трактуя результат как единые (простые) данные, описывает массив ${}^{n_1,...,n_k}D^{f_x}$ (в частном случае вектор ${}^{n}D^{f_x}$), который будем называть массив записей. Массивы записей именуются аналогично массивам простых данных в виде $MS\ [n_1,...n_k], VS\ [n]$, а доступ к элементам записи осуществляется с помощью пары идентификаторов $MS[j_1,...,j_k]\_X_i$, разделенных символом "_". Доступ к вложенным записям осуществляется с помощью последовательности идентификаторов.

Записи и массивы записей отнесем к классу составных структур данных, а по поводу массивов записей будем утверждать следующее.

Утверждение 2. Массивы записей обладают свойствами заданными утверждением 1.

Доказательство. При трактовке носителя данных D^{f_x} как элементарного доказательство становится тривиальным, так как реализуется в результате использования приведенных выше рассуждений.

Теперь рассмотрим данные, непоследовательно расположенные в памяти, которые будем записывать, используя в качестве разделителя "/", в виде $X_1/.../X_k$.

Здесь уместно сделать небольшое отступление.

Данные бывают связаны с точки зрения последовательности их обработки. Для регулярно организованных данных, в частности массивов, эта связь легко учитывается в силу их последовательного размещения. Для непоследовательно расположенных данных такая связь обеспечивается указателями.

Отметим, что отсутствующие на текущем этапе разработки алгоритма связи между данными могут проявиться на следующих её этапах.

Для непоследовательно расположенных связанных данных введем следующую операцию.

Операция объединения любых непоследовательно расположенных в памяти связанных поименованных данных $X_1 \bigcup X_2 = Y$ (в общем случае $X_1 \bigcup ... \bigcup X_k = Y$) описывает новые данные Y, которые именуются с помощью идентификатора и называются сложными структурами данных. Доступ элементам сложной структуры данных возможен как посредством идентификаторов, так и с помощью указателей. Сложные структуры данных могут являться элементами таких структур.

Исходя из определения операции, приведем следующее свойства сложной структуры данных Y:

$$Y \cup X = Y_1, \tag{8}$$

где X – структура данных, Y_1 – новая сложная структура данных;

$$Y = X_1;...;X_n, \tag{9}$$

где X_i – сложная или произвольная структура данных. Свойство (9) выполняется при условии сохранения связей между данными.

Любые произвольно, в частности непоследовательно, расположенные не связанные данные, которые в результате именования образуют совокупности данных, будем записывать в виде $Y = X_1 | ... | X_k$,. Доступ к элементам совокупностей данных осуществляется посредством их идентификаторов.

Наконец, введем операцию **совмещения данных** $X_i \bigcap X_j$, которая в однозначное соответствие одному адресу памяти ставит два (или более) идентификатора, обеспечивая, таким образом, возможность различной интерпретации одних и тех же данных.

Таким образом, построена алгебра $\langle D;S \rangle$, где D – множество поименованных данных, S – сигнатура, определенных на них операций.

Представление любых данных через образующие элементы $\langle D;S \rangle$ называется схемой данных (СД).

Расширяя посредством предложенной алгебры данных САА\Д, получаем трехосновную алгоритмическую систему – <U;L;D;S>, где U – множество Д-операторов, L – множество логических условий, D – множество данных, S – сигнатура операций, состоящая из операций S_1, принимающих значения на множестве Д-операторов U, логических операций S_2, принимающих значения на множестве логических условий L, и S_3, принимающих значения на множестве данных D.

Покажем возможность решения поставленной задачи в рамках построенного алгебраического аппарата.

Разработка КС алгоритмов

Рассмотрение процесса разработки КС начнем с возможности реализации комплексного подхода к их описанию.

Для этого рассмотрим процесс описания управляющих структур и структур данных, полагая, что алгоритм (фрагмент алгоритма) задан в виде Д-оператора $(Y)O(Y')$, на входе и выходе которого специфицированы идентификаторы совокупностей данных.

Первый подход от управления.

Исключив из рассмотрения промежуточные данные, получим следующие свойства данных в КС:

$$D = D_1 \cup D_2 \cup ... \cup D_k; \quad D' = D'_1 \cup D'_2 \cup ... \cup D'_k.$$

Исключив из числа производных Д-операторов такие, что обрабатывают промежуточные данные, декомпозируем Д-оператор и КС запишем в виде $(Y)O(Y') = O_1 * O_2 * ... * O_k$.

После этого Y и Y' совокупности даны, опишем их в виде СД $Y = X_1 | ... | X_n$ и $Y' = X'_1 | ... | X'_n$.

Если все полученные данные соответствуют функциям всех Д-операторов, то есть соответствуют соотношениям (5), то они являются неделимыми и "распределяются" между Д-операторами $O_1, O_2, ..., O_k$. Если некоторые (или все) данные соотношению (5) не соответствуют, или не соответствуют функциям некоторых (или всех) производных Д-операторов, то они делимые и детализуются, то есть записываются в виде СД.

Покажем возможности детализации структур на примерах такой распространенной структуры данных, как строка.

Строка может рассматриваться как вектор простых данных и в этом случае представляет собой СД $STR = STR[n]$.

Строка переменной длины со счетчиком может быть описана с помощью СД вида $STR = sck * STR[n]$, то есть в виде записи, где sck – переменная-счетчик, а доступ к элементам строки осуществляется по правилам доступа к элементам записи.

Строка с управляемой длиной записывается в виде следующей СД $STRD = DSC \bigcup V[n]$, представляющей собой сложную структуру данных, где DSC – дескриптор строки. Элементы СД могут быть детализованы. В данном случае дескриптор строки запишем в виде записи $DSC = MAX * TD * U$ или в виде последовательно расположенных переменных $DSC = MAX; TD; U$, где MAX – максимальная, а TD – текущая длина строки, U – указатель на строку.

Использование операции совмещения данных позволяет реализовать запись с вариантами $DSC = MAX * TD * (U_1 \cup U_2)$. В этом случае выбор варианта, то есть одного из возможных счетчиков, определяется выбором соответствующей пары имен DSC_U_1 или DSC_U_2.

Сверх того, один дескриптор может использоваться с несколькими строками $STRD3 = DSC \bigcup V_1[n] \bigcup V_2[n] \bigcup V_3[n]$.

Указатели позволяют реализовать списковое представление строк. В простейшем случае строки могут быть представлены однонаправленным списком. В этом случае они описываются следующей СД $STR = STRSP_1 \cup STRSP_2 \cup ... \cup STRSP_k$, компоненты которой при детализации представляет собой СД вида $STRSP_i = STR_i[n]; U$, где $STR_i[n]$ – звено (многосимвольная группа, подстрока), описанное с помощью вектора, или символ, U – указатель на следующий элемент списка (следующую подстроку).

Процесс детализации продолжается до тех пор, пока не достигает уровня, который определяется функциями производных Д-операторов и выполнением соотношения (5).

Отметим, что приведенные примеры, не отражая всего многообразия структур данных, демонстрируют достаточные возможности для их описания. То есть, средствами алгебры данных могут быть описаны разнообразные структуры данных и осуществлена их детализация. В частности, вектор может интерпретироваться как дек, стек, очередь, а указатели позволяют описывать списки, деревья и т. п. Кроме того, указатели, ссылающиеся на указатели, образуют указатели высшего ранга [10], они могут быть организованы в массивы и использоваться для описания сложных структур данных.

После того, как данные были детализованы и специфицированы на входах и выходах производных Д-операторов, КС дополняется Д-операторами, обрабатывающими промежуточные данные, на входе и выходе которых эти данные специфицируются. При этом промежуточные данные, если они делимые, описываются в виде СД и таким образом детализуются и специфицируются на входах и выходах производных Д-операторов аналогично вышеописанному случаю.

На любом следующем (или предыдущем) этапе разработки может быть использован подход – от данных. Отличие в данном случае состоит в том, что процесс разработки начинается с детализации данных. После этого исходный Д-оператор декомпозируется таким образом, что производные Д-операторы обрабатывают полученные данные.

Очевидно, что на каждом этапе могут использоваться оба подхода одновременно. По-видимому, при разработке программ, интуитивно используется именно такой вариант.

В этом случае процесс описания алгоритма разбивается на последовательность шагов. На каждом шаге сначала выделяется Д-оператор и для него детализуются данные или наоборот шаг начинается с детализации данных. В результате записывается Д-оператор $(X_i)O_i(X_i')$ со специфицированными данными такими, что $X_i^{pr} \subseteq X_i$ и $X_i'^{pr} \subseteq X_i'$, если промежуточные данные имеют место. На каждом очередном шаге операции повторяются в произвольном порядке, то есть вводится Д-оператор $(X_j)O_j(X_j')$, причем данные на его входе и выходе специфицируются с учетом имеющих место промежуточных данных. Таким образом, согласование данных с управляющей структурой происходит на каждом шаге, что упрощает и повышает качество этого процесса. Кроме того, облегчается работа с промежуточными данными.

Таким образом, при описании КС алгоритмов может быть использован комплексный подход, при котором описание данных и управляющих структур осуществляется согласованно.

При использовании любых подходов, стратегий и методов разработки алгоритмов, как правило, возникает задача их трансформации, если получен неудовлетворяющий результат или в случае модернизации алгоритма. Возможность трансформации данных в этом случае обеспечивается соотношениями (6) – (9) и утверждениями 1,2.

Одним из важнейших требований, предъявляемых к формальному аппарату, является обеспечение возможности контроля за корректностью описания и обработки данных. Рассмотрим такие возможности.

Очевидным средством контроля за корректностью описания данных в рамках алгебры данных, является проверка выполнения соотношений (2) – (4). Отметим, что этими соотношения свойства данных не исчерпываются, но в данной работе другие соотношения и, соответственно, средства контроля не рассматриваются.

Действенным средством контроля за корректностью обработки данных в программировании является их типизация. Этот механизм может быть использован и на алгоритмическом уровне.

Тип данных определим следующим образом.

Определение 4. Типами данных, образующими множество T, будем называть интерпретации данных, хранимых в некотором формате $f_x \in F$, которые определяют множество операций, применимых к этим данным. Если для данных в некотором формате определено множество операций, то они называются типизированными. В противном случае – не типизированными.

Заметим, что данные в одном формате могут интерпретироваться по-разному и, таким образом, иметь различные типы. Кроме того, если операции определенные над некоторыми структурами данных, например массивами, не полиморфны, то есть зависят от типа образующих его элементов, то для каждого типа данных, образующих массив, определяется свой тип массива.

Таким образом, типизированным данным в соответствие будет поставлено некоторое множество допустимых на них операций. На некотором этапе разработки, когда на входе Д-оператора специфицированы эти и только эти данные, а Д-оператор выполняет над данными одну операцию, то достаточно легко, при этом автоматически, проверить входит ли эта операция в множество допустимых.

Заключение

В результате выполненных исследований построена трехосновная алгебраическая система и показана возможность реализации комплексного подхода к программированию при описании КС алгоритмов и возможность согласованности в описании управляющих структур и структур данных.

Описание структур данных в виде СД позволяет их детализовать, обеспечивая спецификацию данных на входе и выходе производных Д-операторов. Структуры данных могут быть реконфигурированы в соответствии с изменением алгоритма решаемой задачи. Кроме того, предложены некоторые средства контроля за корректностью описания и обработки данных.

Перспективным направлением развития полученных результатов является описание известных структур данных в виде СД и апробация полученных результатов на конкретных задачах.

1. *Данные* в языках программирования: абстракция и типология. Сб. статей / Под ред. В.Агафонова. – М.: Мир, 1982. – 328 с.
2. *Bastani F.B., Iyengar S.S.* The effect of data structures on the logical complexity of programs // CACM. – 1987. – V. 30, N 3. – P. 250 – 259.
3. *Акуловский В.Г.* Основы алгебры алгоритмов, базирующейся на данных // Проблеми програмування. – 2010. – № 2–3. – С. 89 – 96.
4. *Дорошенко А.Е., Акуловский В.Г.* Алгебра алгоритмов с данными и прогнозирование вычислительного процесса // Проблеми програмування. – 2011. – № 3. – С. 3 – 10.
5. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – К.: Наукова думка, 1978. – 319с.
6. *Вирт Н.* Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
7. *Цейтлин Г.Е.* Алгоритмические алгебры структур данных и многоуровневое проектирование программ // Программирование. – 1986. – № 3. – С. 8 – 16.
8. *Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К.* Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.
9. *Турский В.* Методология программирования. – М.: Мир, 1981. – 264 с.
10. *Ющенко Е.Л.* Адресное программирование. – Киев: Гостехиздат УССР, 1963. – 288 с.