ИЗВЕСТИЯ
ПЕНЗЕНСКОГО ГОСУДАРСТВЕННОГО
ПЕДАГОГИЧЕСКОГО УНИВЕРСИТЕТА
имени В. Г. БЕЛИНСКОГО
ФИЗИКО-МАТЕМАТИЧЕСКИЕ И ТЕХНИЧЕСКИЕ НАУКИ
№ 18 (22) 2010

IZVESTIA
PENZENSKOGO GOSUDARSTVENNOGO
PEDAGOGICHESKOGO UNIVERSITETA
imeni V. G. BELINSKOGO
PHYSICAL, MATHEMATICAL AND TECHNICAL SCIENCES
№ 18 (22) 2010

УДК 004.6

КОНСТРУКТИВНЫЕ СТРУКТУРЫ ДАННЫХ

© А. М. ВОЛОДИН
Пензенский государственный педагогический университет им. В. Г. Белинского,
кафедра прикладной математики и информатики
e-mail: a.m.volodin@gmail.com

Володин А. М. – Конструктивные структуры данных // Известия ПГПУ им. В. Г. Белинского. 2010. № 18 (22). С. 118–122. – *Предлагается рассматривать структуру данных как конструктивную систему с двумя уровнями организации: логической и физической и системно-изоморфным соответствием между ними. Конструктивная структура данных является функциональной системой, способной самостоятельно формировать и поддерживать корректное отображение логической структуры данных в физические структуры данных. Приведена архитектура конструктивной структуры данных и примеры реализации логической структуры данных различными физическими структурами данных.*

Ключевые слова: структура данных, логическая организация данных, физическая организация данных, конструктивная структура данных, изоморфизм, системный изоморфизм, топология структур данных.

Volodin A. M. – Constructive data structures // Izv. Penz. gos. pedagog. univ. im.i V. G. Belinskogo. 2010. № 18 (22). P. 118–122. – *It is offered to consider data structure as a constructive system with two levels of organization: logical and physical and system-isomorphic correspondence between them. Constructive data structure is a functional system that can independently form and support proper mapping of logical data structure to physical data structures. The architecture of constructive data structure and examples of implementation of logical data structure with different physical data structures are given.*

Key words: data structure, logical data organization, physical data organization, constructive data structures, isomorphism, system isomorphism, topology of data structures.

Структура данных (СД) определяется как совокупность элементов данных и отношений между ними. Это определение СД отражает только логическую организацию данных (ЛОД), которая должна быть зафиксирована любым подходящим способом в физической организации данных при разработке программного обеспечения.

Физическая организация данных (ФОД) также представляется некоторой СД, которая носит конкретный характер и определяет размещение данных в памяти, методы доступа к данным и способы реализации операций обработки данных.

Поэтому целесообразно рассматривать СД как конструктивную систему с двумя уровнями организации данных: логической и физической, на которых используются СД определенного типа, связанные отношением соответствия. СД такого типа будем называть конструктивной структурой данных (КСД).

КСД обладает свойством «разложимости», т.к. каждый ее уровень отражает определенные особенности организации данных и обладает достаточной обособленностью. Это придает КСД большую автономность и устойчивость.

Для СД определены следующие уровни организации: конструктивный элемент, узел, подструктура, структура. При этом будем учитывать наличие двух типов структур данных: абстрактных (логических), ориентированных на адекватное отображение реального мира, и конкретных (физических), обеспечивающих обработку данных на компьютере.

Базовыми (конструктивными) элементами для построения СД являются элементы данных и указатели, а также дополнительная информация, представленная в виде параметров и признаков [1]. Конструктивные элементы являются минимальными компонентами СД. Они могут использоваться на уровнях элемента данных, узла, подструктуры и СД в целом.

Конструктивные элементы логических СД представляют собой абстрактные элементы, для которых может быть задан абстрактный тип данных.

На физическом уровне конструктивные элементы задаются с помощью конкретных (физических) элементов, для которых определен тип данных, реализуемый на компьютере (в языке программирования).

Сильно связанные элементы группируются в узлы (блоки). Например, узлы бинарного дерева

содержат элемент данных и два указателя на узлы-последователи и могут содержать указатель на предшествующий узел. Аналогично в узлах m-арного дерева (например, B-дерева) могут храниться не более m-1 элементов данных и не более m указателей.

Узлом называется сильно связанная система (композиция) конструктивных элементов, обычно строящаяся по единым законам и предоставляющая единый метод доступа к элементам.

В композиции узлов могут быть включены параметры, что позволяет строить СД из узлов с одинаковой структурой, но с различным количеством элементов, и признаки, позволяющие формировать гетерогенные СД из узлов с различной структурой.

Структура данных – это система, строящаяся по определенным законам из конструктивных элементов, узлов и подструктур.

Подструктура – это система, являющаяся структурно-функциональной подсистемой сложной структуры данных и обладающая свойствами целостности (рассматривается и используется как элемент СД) и локальности (реализует СД более низкого уровня и операции ее обработки).

КСД является сложной СД, минимально включающей два уровня организации данных: логический и физический, между которыми могут находиться некоторые промежуточные уровни, обеспечивающие более плавный переход от верхнего логического уровня к нижнему физическому уровню. При этом логическая СД не сразу отображается в конкретную физическую СД, а многократно реализуется логическими и физическими СД, постепенно преобразуясь в конкретную физическую СД.

Определение КСД полностью соответствует понятию иерархической системы [2], что позволяет приписать ей ряд существенных свойств иерархических систем. Действительно, описание организации данных на логическом уровне соответствует макроскопическому уровню иерархической системы, а на физическом уровне – микроскопическому.

На макроскопическом (логическом) уровне КСД описывается небольшим числом макроскопических переменных, определяющих элементы данных и отношения между ними. Макроуровень является абстрактным и функциональным, т.к. на этом уровне определяется логика поведения КСД и вырабатываются требования к нижележащим уровням.

Микроскопический (физический) уровень – «энергетически структурный» [2] – представляет реализацию функциональных требований макроуровня. Для реализации ЛОД используются различные физические СД и их композиции: одномерные и многомерные массивы, односвязные, двусвязные и многосвязные списки, файлы с различной организацией данных и др. При этом ФОД может содержать значения данных, сжатые определенным методом, а также включать дополнительные (управляющие) подструктуры, способствующие повышению эффективности обработки данных. Разнообразие ФОД для одной ЛОД влияет только эффективность обработки данных и не изменяет логику обработки СД, т.к. отношения между данными, определенные в логической СД, полностью поддерживаются в физической СД.

Описание ФОД задается большим числом переменных, поскольку учитывает большее число различных физических характеристик. Структуры ФОД могут содержать также дополнительную (служебную или управляющую) информацию, которая не будет видна извне, однако может играть существенную роль при обработке запросов.

Как установлено в [1], ФОД целесообразно поддерживать в виде нерегулярной СД, т.к. только нерегулярные СД способны эволюционировать и улучшать свои характеристики. Это создает им условия для более длительного устойчивого существования в изменяющейся среде.

Причинами возникновения нерегулярности СД являются неравномерность использования данных и специфика их обработки, фиксируемые в дополнительной информации. Анализ этой информации и выявление закономерностей позволяет КСД настраиваться на требуемое использование и обработку данных [1].

Создание систем, способных на основе ЛОД самостоятельно формировать и поддерживать корректное отображение логических СД в физические СД, определяется рядом факторов:

– относительной обособленностью (самостоятельностью, независимостью) и неполной адекватностью (частичным соответствием) логических и физических СД;

– возможностью включения дополнительной информации в физическую СД, повышающую эффективность представления и обработки логической СД;

– снижением эффективности, увеличением времени обработки и потерей данных, т.е. деградация физической СД, при увеличении объема и количества связей между элементами данных;

– сложностью и трудоемкостью разработки алгоритмов обработки данных при увеличении сложности физической СД и др.

Данную проблему целесообразно решать на основе системного подхода, рассматривающего ЛОД и ФОД как достаточно обособленные системы, между которыми устанавливается определенное соответствие.

В [3] рассмотрена проблема соответствия концептуальной модели предметной области и схемы БД и показано, что между ними целесообразен системный изоморфизм, при котором концептуальная модель предметной области полностью отображается в схему БД, а схема БД частично отображается в концептуальную модель. Этот же принцип может быть использован для обоснования соответствия ЛОД и ФОД, при котором структуры ЛОД полностью отображаются в структуры ФОД, а структуры ФОД частично отображаются в структуры ЛОД. Это означает, что на логическом уровне КСД позволяет в полном объеме получать и обрабатывать “видимые” извне данные, а на физическом уровне обеспечивает разнообразие ФОД как за счет использования различных способов организации данных, так и дополнительной информации. Поэтому

существует множество альтернативных реализаций ЛОД, обеспечивающих различную эффективность обработки данных.

Системный изоморфизм, не являющийся статическим, обеспечивает необходимое подобие ЛОД и ФОД в КСД. Каждая из структур может изменяться относительно независимо (эволюционировать), а КСД должна самостоятельно поддерживать динамическое системно-изоморфное соответствие между ними. Это позволит создавать и поддерживать ЛОД, обеспечивающую высокий уровень адекватности внешней среде, а также формировать и поддерживать ФОД, максимально ориентированную на надежную и эффективную организацию и обработку данных.

Для наглядного представления СД удобно использовать графы, вершины которого представляют элементы (узлы, содержащие элементы данных, параметры и признаки), а дуги – отношения (связи, указатели) между элементами. Для физических СД целесообразно задавать метки (функциональные символы), символизирующие типы элементов и отношений между ними.

СД можно рассматривать как топологическое пространство, заданное множеством элементов и отношением соседства между ними. Отношение соседства характеризует способ перехода от одного элемента к другому и, соответственно, определяет метод доступа к данным. Так, доступ к элементам односвязного списка осуществляется последовательно, начиная с начала списка. Элементы двусвязного списка можно последовательно просматривать в прямом или обратном порядке.

Узлы деревьев связаны отношением иерархии от корня, являющегося входом в структуру, до конечных узлов – листьев, не имеющих потомков. Просмотр элементов возможен только от родительских узлов к узлам-потомкам. Иерархические СД позволяют организовать индексно-последовательный (для упорядоченного множества данных) или индексно-прямой метод доступа (для неупорядоченного множества данных).

Обход графа – это перечисление его вершин и/или дуг от некоторой исходной до некоторой конечной вершины. Алгоритмы обхода (поиск в глубину и ширину) основаны на использовании информации о смежности вершин.

Поскольку для элементов множества определено свойство различимости и не задано ни одного структурного отношения, каждый элемент множества может быть соседним для любого другого, отличного от него, элемента. Произвольный элемент мультимножества является соседним для любых других, необязательно различимых элементов.

Отношение соседства в СД с отсутствием явно заданных связей между элементами, выражено более имплицитно. Например, реализация отношения соседства в массивах основано на преобразовании индекса (позиции) элемента в адрес физической памяти. Поэтому массивы предоставляют прямой доступ к элементам.

Данные разных типов могут объединяться в блоки (записи). Доступ к таким данным осуществляется по их идентификатору (имени).

Для доступа к данным часто применяются словари, в которых используется принцип ассоциативной памяти. В этом случае доступ к данным осуществляется по ключу, принимающему значения из некоторого множества. При реализации словаря хеш-таблицей используется функция преобразования ключа в адрес, по которому можно найти требуемый элемент данных.

Учитывая, что ЛОД может быть реализована на физическом уровне различными способами, КСД имеет сложную топологию, способную изменяться (эволюционировать) в процессе существования КСД. В отдельных ее подструктурах (фрагментах), на разных уровнях иерархии можно увидеть различные топологии, соответствующие различным СД.

Однако не все возможные комбинации элементов являются допустимыми, способными объединить структуры различных иерархических уровней в одну сложную систему. Основной принцип соединения частей в целое состоит в установлении общего темпа их эволюции (коэволюции) [4]. Топологически правильное объединение – это объединение элементов структуры в соответствии с тенденциями организации (потребностями) внешней среды.

Наиболее распространенными логическими СД являются множество, список, список с пропусками, кольцевой список, бинарное дерево, m-арное дерево, сеть и граф общего вида.

В качестве примера рассмотрим реализацию списка, приведенного на рис. 1.

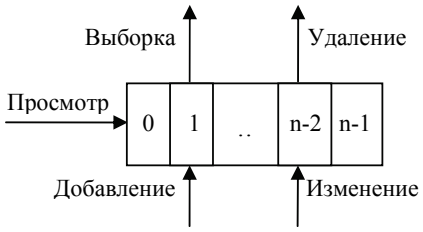
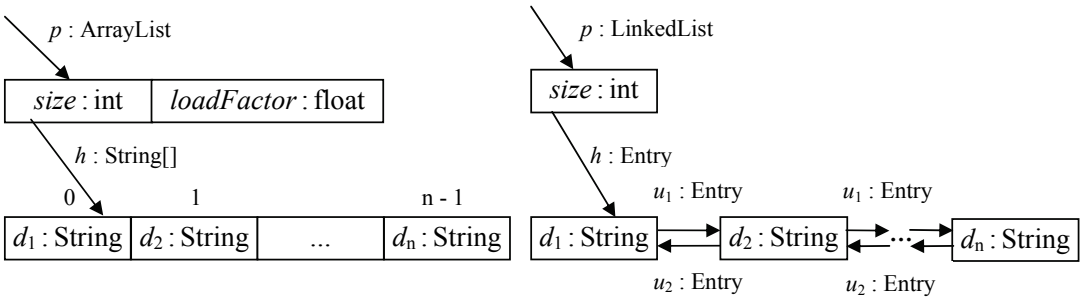


Рис. 1. Абстрактная СД "список".

Список относится к классу последовательностей и является простой, с точки зрения реализации, логической СД, в которой определено отношение линейного порядка, задающее для каждого его элемента всего два соседних: предыдущий и следующий. Просмотр элементов осуществляется последовательно, начиная с начала списка. Операции выборки, изменения и удаления могут производиться над любыми элементами списка. Добавление нового элемента может осуществляться в любое место списка в соответствии с заданным отношением порядка.

На рис. 2 приведены примеры возможной реализации списка с помощью одномерного массива и двусвязного списка.



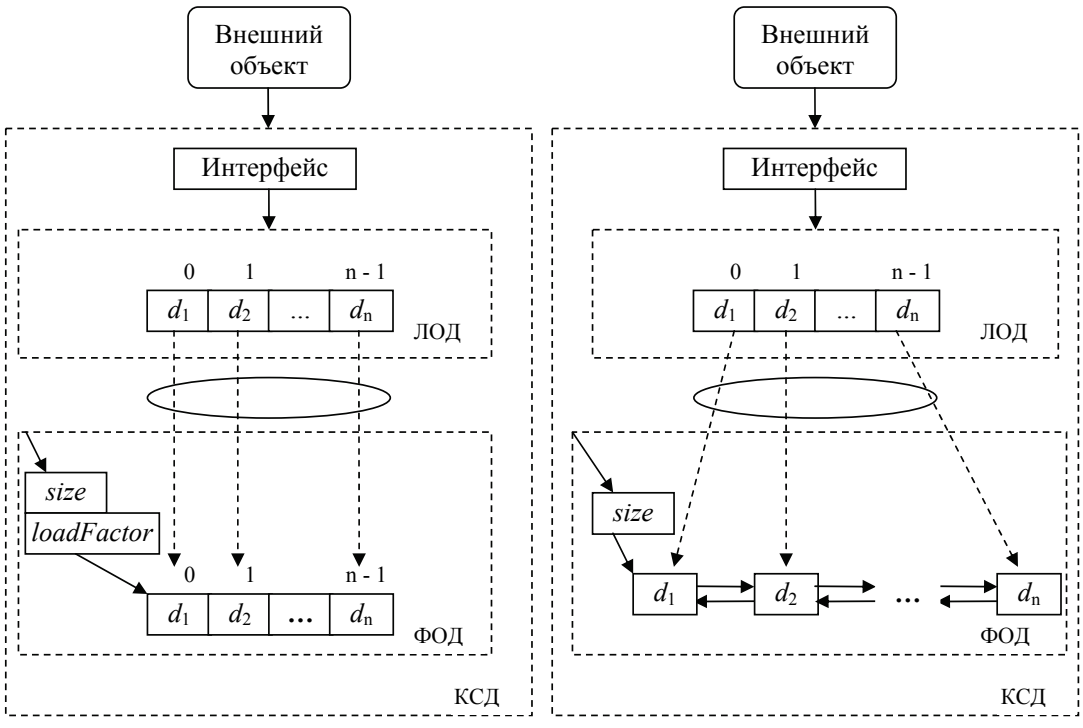
а) Реализация списка одномерным массивом, где p – указатель на список, ArrayList – метка, обозначающая список, реализованный массивом, h – указатель на массив, String[] – метка, обозначающая массив, int, float, String – метки, символизирующие целый, вещественный и строковый тип данных, d_i – значение i-го элемента данных, i – индекс (позиция элемента в массиве), *size* – признак, содержащий число заполненных ячеек массива, *loadFactor* – коэффициент заполнения массива (отношение числа заполненных ячеек к общему числу ячеек массива) – параметр, задающий условие изменения размерности массива.

б) Реализация списка двусвязным списком, где p – указатель на список, LinkedList – метка, обозначающая связный список, h – указатель на начало списка, u_1, u_2 – указатели на следующий и предыдущий элементы списка, Entry – метка, обозначающая узел списка, int, String – метки, символизирующие целый и строковый тип данных, d_i – значение i-го элемента данных, *size* – признак, содержащий количество элементов в списке.

Рис. 2. Реализации списка.

Массив обеспечивает быстрый (прямой) доступ к элементам, однако медленно добавляет и удаляет их, а двусвязный список предоставляет медленный (последовательный) доступ к элементам, но быстро их добавляет и удаляет. Поэтому если в приложении преобладают операции добавления или удаления элементов целесообразно использовать реализацию списка связным списком (односвязным или двусвязным). Если доминируют операции получения прямого доступа к элементам, то – одномерный массив.

На рис. 3 приведена схема соответствия абстрактной СД “список”, составляющей ЛОД КСД, физическим СД, реализованным одномерным массивом и двусвязным списком, составляющим ФОД КСД. Каждый элемент списка отображается в соответствующий элемент физической СД, которая содержит также дополнительную информацию.



а) б)

Рис. 3. КСД с разной ФОД.

Внешние объекты взаимодействуют с КСД через интерфейс, предоставляющий доступ к логическому уровню организации данных, который является виртуальным (воображаемым). Реальная организация данных и их обработка осуществляются на физическом уровне, который в полном объеме поддерживает логическое представление данных. Поэтому нет необходимости организовывать и хранить данные на логическом уровне, а достаточно формировать их на основе отображения ЛОД-ФОД.

Интерфейс доступа к данным в КСД представляется в виде предметно ориентированного языка (Domain Specific Language, DSL), специфицирующего логическую СД и методы ее обработки.

КСД – это открытая функциональная система [5], существующая в определенной среде и обменивающаяся с ней «веществом», «энергией» и «информацией» [1]. В роли внешней среды выступает программа, в которой реализована и используется КСД.

Основными принципами организации КСД являются корректность, надежность, устойчивость, эффективность, адаптивность, распределенность, способность к эволюции, открытость, непрерывность функционирования, живучесть и др. Реализация этих принципов повышает надежность и эффективность функционирования КСД в непрерывно изменяющейся внешней среде.

Причиной активности и источником развития функциональных систем являются отклонения параметров от нормы при взаимодействии системы с внешней средой [5]. Благодаря отклонениям изменяются внутрисистемные связи, возрастает способность системы отражать и реагировать на изменения внешней среды, следовательно, формируется обратная связь.

КСД как функциональная система в процессе своего существования постоянно совершенствует ФОД. В качестве отклонений здесь выступает изменение типа и интенсивности запросов на обработку данных, что позволяет обоснованно изменять физическую СД и обеспечивать более эффективную реализацию соответствующих запросов.

Операции обработки данных предполагают определенную последовательность просмотра элементов СД, поэтому для повышения эффективности функционирования КСД необходимо минимизировать число обращений к данным. Это требует изменения топологии структур ФОД. Например, целесообразно устанавливать отношение соседства между элементами, удовлетворяющими критериям поиска. Полученные отношения соседства в дальнейшем необходимо учитывать при преобразовании ФОД.

Таким образом, затраты на поддержание динамического соответствия структур ЛОД и ФОД будут компенсироваться повышением адекватности СД внешней среде и высокой эффективностью обработки данных, а также создавать условия для возникновения самоорганизации.

Эволюция КСД представляет собой необратимую последовательность допустимых изменений структуры во времени. Для обеспечения эволюции КСД целесообразно реализовать в виде автономного компонента организации данных (АКОД), способного самостоятельно функционировать в среде самоорганизующейся информационной системы [6]. КСД в форме АКОД позволяет реализовать все возможности СД с многоуровневой организацией данных и создать объект, осуществляющий хранение и обработку данных и способный к совершенствованию и развитию в процессе своего существования.

СПИСОК ЛИТЕРАТУРЫ

1. Дрождин В.В., Володин А.М. Синергетический подход к организации структур данных // Программные продукты и системы. 2010. № 1. С. 29–34.
2. Николис Дж. Динамика иерархических систем: эволюционное представление. М.: Мир, 1989. 488 с.
3. Зинченко Р.Е. Системно-изоморфное динамическое соответствие концептуальной модели предметной области и схемы базы данных // Программные продукты и системы. 2010. № 1. С. 71–75.
4. Князева Е.Н., Курдюмов С.П. Синергетика: нелинейность времени и ландшафты коэволюции. М.: КомКнига, 2007. 272 с.
5. Абдеев Р.Ф. Философия информационной цивилизации. М.: ВЛАДОС, 1994. 336 с.
6. Дрождин В.В., Володин А.М. Автономный компонент организации данных // Проблемы информатики в образовании, управлении, экономике и технике. Сб. стат. VIII Всеросс. науч.-технич. конф. Пенза: ПГПУ, 2008. С. 7–14.