Уральский федеральный университет

СТРУКТУРЫ ДАННЫХ

Лектор
Спиричева Наталия Рахматулловна

Ст. преподаватель каф. ИТ
Ауд. Р-246

Структуры данных

Составитель курса лекций:
Спиричева Наталия Рахматулловна,
ст. преподаватель каф. Информационных технологий

Общие характеристики, классификация и особенности применения типов и структур данных

Цели изучения

- Изучение основных характеристик алгоритма;
- Знакомство классификацией структур данных.

Основная литература

1. Вирт. Н Алгоритмы и структуры данных– Москва : ДМК-Пресс, 2010 — 272 с.
2. Котов В.М. Алгоритмы и структуры данных. –Минск: БГУ,2011.-267с
3. Гагарина Л.Г. Алгоритмы и структуры данных. Москва: ИНФРА-М, 2009 – 304 с.
4. Т. Х. Кормен, Ч.И. Лейзерсон, Р.Л. Ривест, К. Штайн. Алгоритмы: построение и анализ. – Вильямс, 2006
5. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы. – Вильямс, 2000
6. Кнут Д. Искусство программирования для ЭВМ. (1978)

Содержание

Основные темы лекции:

- Концепция структур данных и алгоритмов их обработки
- Классификация структур данных
- Операции над структурами данных
- Структурность данных и технология программирования

Тема 1: Концепция структур данных и алгоритмов их обработки

(Н. Вирт)

Концепция структур данных и алгоритмов их обработки

- Структуры данных и алгоритмы служат теми материалами, из которых строятся программы. Более того, сам компьютер состоит из структур данных и алгоритмов.

- Встроенные структуры данных представлены теми регистрами и словами памяти, где хранятся двоичные величины. Заложенные в конструкцию аппаратуры алгоритмы - это воплощенные в электронных логических цепях жесткие правила, по которым занесенные в память данные интерпретируются как команды, подлежащие исполнению.

Концепция структур данных и алгоритмов их обработки

Алгоритм – свод конечного числа правил, задающих последовательность выполнения операций при решении той или иной специфической задачи.

Само по себе слово “алгоритм” (algorithm) очень интересно. Это слово еще не вошло в Webster’s New World Dictionary даже 1957 года издания. Там можно найти только старую форму “Algorism”, что означает правило выполнения арифметических действий с использованием арабских цифр. К средним векам сложились две враждующие партии среди приверженцев различных традиций счета. Абакисты считали на абаках – счетах, алгоритмики же использовали начатки математической символики. Происхождение слова algorism долгое время оставалось неясным.

algorism: от имени автора известного арабского учебника по математике – Abu Ja’far Mohammed ibn Musa al-Khowarizmi (около 825 г.), означающего буквально “Отец Джафара, Магомет, сын Моисея, уроженец Ховаризма”. В настоящее время Ховаризм – город Хива. Вышеназванный уроженец Ховаризма написал знаменитую книгу “Kitab al jabr w’al-muqabala” (“Правила восстановления и преобразования”); заглавие этой книги дало начало другому слову – “алгебра”, хотя сама книга в действительности была не совсем алгебраической.

Постепенно форма и значение слова “algorism” исказились; как объясняет “Oxford English Dictionary”, слово было “ошибочно видоизменено” в результате “укоренившейся путаницы” со словом arithmetic. Изменение algorism на algoritm нетрудно понять, если учесть, что истинное происхождение слова давно было забыто.

Алгоритм Евклида

К 1950 г. под словом алгоритм чаще всего подразумевали изложенный Евклидом процесс нахождения наибольшего общего делителя двух чисел (алгоритм Евклида).

Алгоритм Евклида

Даны два целых положительных числа m и n. Требуется найти их наибольший общий делитель, т.е. наибольшее положительное целое число, которое нацело делит как m, так и n.

- Шаг 1. [Нахождение остатка.] Разделим m на n. Пусть остаток равен r. (Имеем 0<=r<n.)
- Шаг 2. [Это нуль?] Если r=0, алгоритм кончается; n – искомое число.
- Шаг 3. [Замена] Положите m<-n, n<-r и возвращайтесь к шагу 1.

Алгоритм имеет пять важнейших особенностей:

- Конечность
- Определенность
- Ввод
- Вывод
- Эффективность

- ***Конечность***. Алгоритм должен заканчиваться после конечного числа шагов.
- ***Определенность*.** Каждый шаг алгоритма должен быть точно определен. Действия, которые необходимо произвести, должны быть строго и недвусмысленно определены в каждом возможном случае.
- ***Ввод*.** Алгоритм имеет некоторое (может быть, равное нулю) число входных данных, то есть величин, заданных ему до начала работы. Эти данные берутся из некоего конкретного множества объектов.

- **Вывод.** Алгоритм имеет одну или несколько выходных величин, то есть величин, имеющих вполне определенные отношения ко входным данным.
- **Эффективность.** От алгоритма требуется также, чтобы он был эффективным. Это означает, что все операции, которые необходимо произвести в алгоритме, должны быть достаточно простыми, чтобы их в принципе можно было выполнить точно и за конечный отрезок времени с помощью карандаша и бумаги.

На практике нам нужны *хорошие* алгоритмы.

Они определяются характеристиками:

1. число, указывающее, сколько раз выполняется каждый шаг алгоритма.
2. приспособляемость алгоритма к вычислительным машинам,
3. простота
4. изящество

Структура данных относится, по существу, к "пространственным" понятиям: ее можно свести к схеме организации информации в памяти компьютера. Алгоритм же является соответствующим процедурным элементом в структуре программы - он служит рецептом расчета.

Тема 2: Классификация структур данных

- Понятие "ФИЗИЧЕСКАЯ структура данных" отражает способ физического представления данных в памяти машины и называется еще структурой хранения, внутренней структурой или структурой памяти.

- Рассмотрение структуры данных без учета ее представления в машинной памяти называется абстрактной или ЛОГИЧЕСКОЙ структурой.

- Различаются ПРОСТЫЕ (базовые, примитивные) структуры (типы) данных и ИНТЕГРИРОВАННЫЕ (структурированные, композитные, сложные).

- Простыми называются такие структуры данных, которые не могут быть расчленены на составные части, большие, чем биты. С точки зрения физической структуры важным является то обстоятельство, что в данной машинной архитектуре, в данной системе программирования мы всегда можем заранее сказать, каков будет размер данного простого типа и какова структура его размещения в памяти. С логической точки зрения простые данные являются неделимыми единицами.

- Интегрированными называются такие структуры данных, составными частями которых являются другие структуры данных - простые или в свою очередь интегрированные.

- В зависимости от отсутствия или наличия явно заданных связей между элементами данных следует различать НЕСВЯЗНЫЕ структуры (векторы, массивы, строки, стеки, очереди) и СВЯЗНЫЕ структуры (связные списки).

- Весьма важный признак структуры данных - ее изменчивость - изменение числа элементов и (или) связей между элементами структуры. В определении изменчивости структуры не отражен факт изменения значений элементов данных, поскольку в этом случае все структуры данных имели бы свойство изменчивости. По признаку изменчивости различают структуры СТАТИЧЕСКИЕ, ПОЛУСТАТИЧЕСКИЕ, ДИНАМИЧЕСКИЕ.

- Важный признак структуры данных - характер упорядоченности ее элементов. По этому признаку структуры можно делить на ЛИНЕЙНЫЕ И НЕЛИНЕЙНЫЕ структуры.

- В зависимости от характера взаимного расположения элементов в памяти линейные структуры можно разделить на структуры с ПОСЛЕДОВАТЕЛЬНЫМ распределением элементов в памяти (векторы, строки, массивы, стеки, очереди) и структуры с ПРОИЗВОЛЬНЫМ СВЯЗНЫМ распределением элементов в памяти (односвязные, двусвязные списки). Пример нелинейных структур - многосвязные списки, деревья, графы.

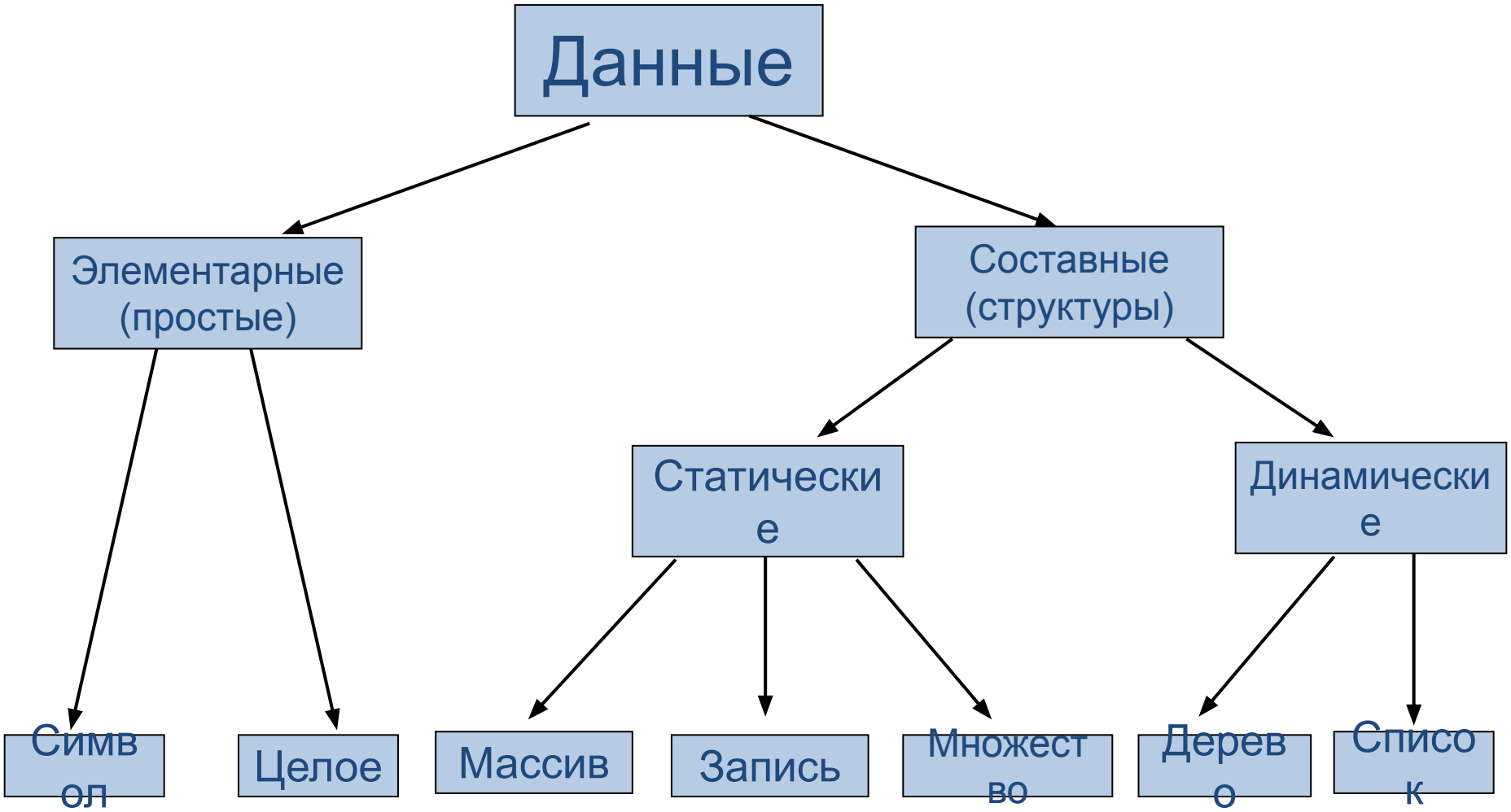
В языках программирования понятие "*структуры данных*" тесно связано с понятием "*типы данных*". Информация по каждому типу однозначно определяет :

- структуру хранения данных указанного типа, т.е. выделение памяти и представление данных в ней, с одной стороны, и интерпретацию двоичного представления, с другой;
- множество допустимых значений, которые может иметь тот или иной объект описываемого типа;
- множество допустимых операций, которые применимы к объекту описываемого типа.

- В большинстве случаев новые типы данных определяются с помощью ранее определенных типов данных. Значения, принадлежащие к такому типу, обычно представляют собой совокупности значений компонент, принадлежащих к определенным ранее типам компонент, такие составные значения называются структурированными. Если имеется только один тип компонент, т.е. все компоненты принадлежат одному типу, то он называется **базовым.**

- Число различных значений, принадлежащих типу T, называется **кардинальным числом** T. Кардинальное число определяет размер памяти, нужной для размещения переменной x типа T. Этот факт обозначается так: **x:T.**

Структуры данных


Данные
Элементарные (простые)
Составные (структуры)
Символ
Целое
Статические
Динамические
Массив
Запись
Множество
Дерево
Список

- Структура данных — это исполнитель, который организует работу с данными, включая их хранение, добавление и удаление, модификацию, поиск и т.д. Структура данных поддерживает определенный порядок доступа к ним. Структуру данных можно рассматривать как своего рода склад или библиотеку.

- При описании структуры данных нужно перечислить набор действий, которые возможны для нее, и четко описать результат каждого действия. Будем называть такие действия **предписаниями**. С программной точки зрения, системе предписаний структуры данных соответствует набор функций, которые работают над общими переменными.

- Структуры данных можно реализовывать и в традиционных языках программирования, и в объектно-ориентированных.

- При этом следует придерживаться объектно-ориентированного стиля программирования: четко выделить набор функций, которые осуществляют работу со структурой данных, и ограничить доступ к данным только этим набором функций. Сами данные реализуются как статические (не глобальные) переменные.

- Структура данных обычно реализуется на основе более простой **базовой структуры**, ранее уже реализованной, или на основе массива и набора простых переменных. Следует четко различать описание структуры данных с логической точки зрения и описание ее реализации. Различных реализаций может быть много, с логической же точки зрения (т.е. с точки зрения внешнего пользователя) все они эквивалентны и различаются, возможно, лишь скоростью выполнения предписаний.

Тип данных — фундаментальное понятие теории программирования. Тип данных определяет множество значений, набор операций, которые можно применять к таким значениям и, возможно, способ реализации хранения значений и выполнения операций. Любые данные, которыми оперируют программы, относятся к определённым типам.

Тип (**сорт**) — относительно устойчивая и независимая совокупность элементов, которую можно выделить во всём рассматриваемом множестве (предметной области).

Математически тип может быть определён двумя способами:

- Множеством всех значений, принадлежащим типу.
- Предикатной функцией, определяющей принадлежность объекта к данному типу

- Типы данных различаются начиная с нижних уровней системы. В Ассемблере x86 различаются типы «целое число» и «вещественное число». Это объясняется тем, что для чисел рассматриваемых типов отводятся различные объёмы памяти, используются различные регистры микропроцессора, а для операций с ними применяются различные команды Ассемблера и различные ядра микропроцессора.

- Концепция типа данных появилась в языках программирования высокого уровня как естественное отражение того факта, что обрабатываемые программой данные могут иметь различные множества допустимых значений, храниться в памяти компьютера различным образом, занимать различные объёмы памяти и обрабатываться с помощью различных команд процессора.

- Как правило, типы в языках программирования не всегда строго соответствуют подобным типам в математике. Например, тип «целое число» большинства языков программирования не соответствует принятому в математике типу «целое число», так как в математике указанный тип не имеет ограничений ни сверху, ни снизу, а в языках программирования эти ограничения есть.

- Теоретически не может существовать языков, в которых отсутствуют типы (включая полиморфные). Это следует из того, что все языки основаны на машине Тьюринга или на лямбда-исчислении. И в том, и в другом случае необходимо оперировать как минимум одним типом данных — хранящимся на ленте (машина Тьюринга) или передаваемым и возвращаемым из функции (лямбда-исчисление).

Языки без типов

Теоретически не может существовать языков, в которых отсутствуют типы. Это следует из того, что все языки основаны на машине Тьюринга или на лямбда-исчислении. И в том, и в другом случае необходимо оперировать как минимум одним типом данных — хранящимся на ленте (машина Тьюринга) или передаваемым и возвращаемым из функции (лямбда-исчисление).

Базовые типы

Каждый язык программирования поддерживает один или несколько встроенных типов данных (базовых типов), кроме того, развитые языки программирования предоставляют программисту возможность описывать собственные типы данных, комбинируя или расширяя существующие.

Преимущества от использования типов данных

- **Надёжность**. Типы данных защищают от трёх видов ошибок
- **Стандартизация**. Благодаря соглашениям о типах, поддерживаемых большинством систем программирования, сложилась ситуация, когда программисты могут быстро менять свои рабочие инструменты, а программы не требуют больших переделок при переносе исходных текстов в другую среду.
- **Документация**. Использование того или другого типа данных объясняет намерения программиста

Можно приводить различные классификации типов данных, например, простые и составные типы, предопределенные и определяемые типы и т.д. Существенно то, что несмотря на многолетнее использование типов данных в отечественном программировании, так и не сложилась устойчивая и общепринятая русскоязычная терминология.

- Встроенные типы данных, т.е. типы, предопределенные в языке программирования или языке баз данных.
- «Уточняемый тип данных" - возможность определения типа на основе встроенного типа данных, значения которого упорядочены.
- Перечисляемые типы данных - явно определяемые целые типы с конечным числом именованных значений.

- Конструируемые типы (иногда их называют составными) обладают той особенностью, что в языке предопределены средства спецификации таких типов и некоторый набор операций, дающих возможность доступа к компонентам составных значений

- Указательные типы дают возможность работы с типизированными множествами абстрактных адресов переменных, содержащих значения некоторого типа.

- Типы, определяемыми пользователями

Контрольные вопросы

- Перечислите особенности алгоритма?
- Дайте определение структуры данных?
- Какие существуют классификации структур данных?

Спасибо за внимание!