АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

ВВЕДЕНИЕ

Они служат базовыми элементами любой машинной программы. В организации структур данных и процедур их обработки заложена возможность проверки правильности работы программы.

Никлас Вирт

Без понимания структур данных и алгоритмов невозможно создать сколько-нибудь серьезный программный продукт. Поэтому главная задача данного учебного курса заключается в следующем:

- показать все разнообразие имеющихся структур данных, представление их в памяти на физическом уровне, т.е., "как это сделано внутри", и на логическом уровне, т.е., как эти структуры реализованы в языках программирования;
- выполняемые над ними операции физического и логического уровней;
- показать значение структурного подхода к разработке алгоритмов, продемонстрировать порядок разработки алгоритмов наиболее интересных задач.

При изучении дисциплины будет приведена классификация структур данных, обширная информация о физическом и логическом представлении структур данных всех классов памяти ЭВМ:

простых, статических, полустатических, динамических;

исчерпывающая информация об операциях над всеми перечисленными структурами.

Изучено большое количество алгоритмов выполнения особенно важных операций, реализованных в виде процедур и функций, которые могут быть применены как "заготовки" в самостоятельных разработках

Тема 1.
Общие сведения о структурах данных

Некоторые понятия и определения

Программируя решение любой задачи, необходимо выбрать ***уровень абстрагирования***.

Иными словами, определить множество данных, представляющих предметную область решаемой задачи.

При выборе следует руководствоваться проблематикой решаемой задачи и способами представления информации.

Здесь необходимо ориентироваться на те средства, которые предоставляют системы программирования и вычислительная техника, на которой будут выполняться программы. Во многих случаях эти критерии не являются полностью независимыми.

Независимо от содержания и сложности любые данные в памяти ЭВМ представляются последовательностью двоичных разрядов, или битов, а их значениями являются соответствующие двоичные числа.

Данные, рассматриваемые в виде последовательности битов, имеют очень простую организацию или, другими словами, слабо структурированы. Для человека описывать и исследовать сколько-нибудь сложные данные в терминах последовательностей битов весьма неудобно.

Более крупные и содержательные, нежели бит, "строительные блоки" для организации произвольных данных получаются на основе понятия "структуры данных".

Определение

Структуры данных - это совокупность элементов данных и отношений между ними.
При этом под ***элементами данных*** может подразумеваться как простое данное так и структура данных.
Под ***отношениями между данными*** понимают функциональные связи между ними и указатели на то, где находятся эти данные.

Элемент отношений - это совокупность всех связей элемента с другими элементами данных, рассматриваемой структуры.

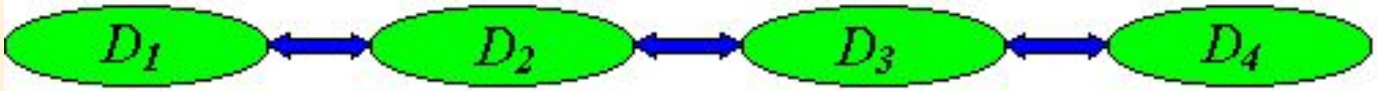
Определение

Такое определение охватывает все возможные подходы к структуризации данных, но в каждой конкретной задаче используются те или иные его аспекты.
Поэтому вводится дополнительная классификация структур данных, направления которой соответствуют различным аспектам их рассмотрения.

Определение

Как бы сложна ни была структура данных, в конечном итоге она состоит из простых данных:

Одномерный массив



Двумерный массив

i / j	1	2	…	n
1	элемент данных	элемент данных	…	элемент данных
2	элемент данных	элемент данных	…	элемент данных
…	…	…	…	
n	элемент данных	элемент данных	…	элемент данных

Отличие понятий структура данных и тип данных

В языках программирования тип данных определяет:

- возможные значения переменных, констант, функций, выражений, принадлежащих к данному типу;
- внутреннюю форму представления данных в ЭВМ;
- операции и функции, которые могут выполняться над величинами, принадлежащими к данному типу.

Структура данных определяет набор переменных, возможно, различных типов данных, объединенных определенным образом.

Уровни представления данных

Мы, занося информацию в компьютер, представляем ее в каком-то виде, который на наш взгляд упорядочивает данные и придает им смысл.
Машина отводит поле для поступающей информации и задает ей какой-то адрес.
Таким образом получается, что мы обрабатываем данные на логическом уровне, как бы абстрактно, а машина делает это на физическом уровне.

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***.

В общем случае между логической и соответствующей ей физической структурами существует различие, степень которого зависит от самой структуры и особенностей той среды, в которой она должна быть отражена.

Последовательность переходов от логической организации к физической:

Поэтому существуют процедуры, осуществляющие отображение логической структуры в физическую и, наоборот, физической структуры в логическую.

Эти процедуры обеспечивают также доступ к физическим структурам и выполнение над ними различных операций, причем каждая операция рассматривается применительно к логической или физической структуре данных.

Абстрактный тип данных – это математическая модель совместно с различными операторами, определенными в рамках этой модели.

Можно разрабатывать алгоритм в терминах абстрактного типа данных (АТД), но для реализации алгоритма в конкретном языке программирования необходимо найти способ представления АТД в терминах типов данных и операторов, поддерживаемых данным языком программирования.
Для представления АТД и используются структуры данных.

Примеры АТД: список, очередь, стек, дек.

Классификация структур данных

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.

Интегрированные структуры данных конструируются программистом с использованием средств интеграции данных, предоставляемых языками программирования.

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***.

Классификация структур данных

Базовые структуры данных, статические, полустатические и динамические характерны для оперативной памяти и часто называются ***оперативными структурами***.

Файловые структуры соответствуют структурам данных для внешней памяти.

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.

Операции над структурами данных

Над любыми структурами данных могут выполняться четыре общие операции:

- ***создание***,
- ***уничтожение***,
- ***выбор*** (доступ),
- ***обновление***.

Эти операции обязательны для всех структур и типов данных.

Для каждой структуры данных могут быть определены операции специфические, работающие только с данной структурой.

Специфические операции рассматриваются при рассмотрении каждой конкретной структуры данных.

Операция создания

Операция создания заключается в выделении памяти для структуры данных. Память может выделяться в процессе выполнения программы или на этапе компиляции.

В ряде языков (например, в С) для структурированных данных, конструируемых программистом, операция создания включает в себя также установку начальных значений параметров, создаваемой структуры.

Например, в Pascal (I: integer), в С (int I) в результате описания типа будет выделена память для соответствующих переменных.

Операция создания

Для структур данных, объявленных в программе, память выделяется автоматически средствами систем программирования либо на этапе компиляции, либо при активизации процедурного блока, в котором объявляются соответствующие переменные.

Программист может и сам выделять память для структур данных, используя имеющиеся в системе программирования процедуры/функции выделения/освобождения памяти.

В объектно-ориентированных языках программирования при разработке нового объекта для него должны быть определены процедуры создания и уничтожения.

Операция создания

Независимо от используемого языка программирования, имеющиеся в программе структуры данных не появляются "из ничего", а явно или неявно объявляются операторами создания структур.
В результате этого всем экземплярам структур в программе выделяется память для их размещения.

Операция уничтожения

Операция уничтожения структур данных противоположна по своему действию операции создания.
Некоторые языки, такие как BASIC, FORTRAN не дают возможности программисту уничтожать созданные структуры данных.
В языках C, Pascal структуры данных, имеющиеся внутри блока, уничтожаются в процессе выполнения программы при выходе из этого блока.
Операция уничтожения помогает эффективно использовать память.

Операция выбора

Операция выбора используется программистами для доступа к данным внутри самой структуры.

Форма операции доступа зависит от типа структуры данных, к которой осуществляется обращение.

Метод доступа - одно из наиболее важных свойств структур, особенно в связи с тем, что это свойство имеет непосредственное отношение к выбору конкретной структуры данных.

Операция обновления

Операция обновления позволяет изменить значения данных в структуре данных.

Примером операции обновления является операция присваивания, или, более сложная форма - передача параметров.

Структурность данных и технологии программирования

Преимущества знания структур данных

{ Позволяет организовать хранение и обработку данных максимально эффективным образом - с точки зрения минимизации затрат как памяти, так и процессорного времени.

{ Возможность структурирования сложного программного изделия.

Структурирование больших программных изделий

Правильное структурирование изделия дает возможность на каждом этапе разработки сосредоточить внимание разработчика на одной обозримой части изделия или поручить реализацию разных его частей разным исполнителям.

Подходы к структурированию больших программных изделий:

I. на основе ***структуризации алгоритмов*** ("нисходящее" проектирование или "программирование сверху вниз"),
II. на основе ***структуризации данных*** ("восходящее" проектирование или "программирование снизу вверх").

Структуризация алгоритмов

Структурируют прежде всего ***действия***, которые должна выполнять программа.

Большую и сложную задачу, стоящую перед проектируемым программным изделием, представляют в виде нескольких подзадач меньшего объема.
Таким образом, модуль самого верхнего уровня, отвечающий за решение всей задачи в целом, получается достаточно простым и обеспечивает только последовательность обращений к модулям, реализующим подзадачи.

Структуризация алгоритмов

На первом этапе проектирования модули подзадач выполняются в виде "заглушек". Затем каждая подзадача в свою очередь подвергается декомпозиции по тем же правилам.
Процесс дробления на подзадачи продолжается до тех пор, пока на очередном уровне декомпозиции получают подзадачу, реализация которой будет вполне обозримой.
В предельном случае декомпозиция может быть доведена до того, что подзадачи самого нижнего уровня могут быть решены элементарными инструментальными средствами (например, одним оператором выбранного языка программирования).

Структуризация данных

Программирование - это обработка данных.

В программах можно изобретать сколь угодно замысловатые и изощренные алгоритмы, но у реального программного изделия всегда есть Заказчик. У Заказчика есть входные данные, и он хочет, чтобы по ним были получены выходные данные, а какими средствами это обеспечивается - его не интересует.

Таким образом, **задачей любого программного изделия является преобразование входных данных в выходные**.

Структуризация данных

Инструментальные средства программирования предоставляют набор базовых (простых, примитивных) типов данных и операции над ними. Интегрируя базовые типы, программист создает более сложные типы данных и определяет новые операции над сложными типами.

Т.о. базовые типы - "кирпичики", из которых создаются сложные типы - "блоки".

Полученные на первом шаге композиции "блоки" используются в качестве базового набора для следующего шага, результатом которого будут еще более сложные конструкции данных и еще более мощные операции над ними и т.д.

В идеале последний шаг композиции дает типы данных, соответствующие входным и выходным данным задачи, а операции над этими типами реализуют в полном объеме задачу проекта.

Нельзя противопоставлять нисходящее проектирование восходящему, придерживаясь одного выбранного подхода.
Реализация любого реального проекта всегда ведется встречными путями, причем, с постоянной коррекцией структур алгоритмов по результатам разработки структур данных и наоборот.

Инкапсуляция как технологический прием структуризации данных

Смысл ее состоит в том, что сконструированный новый тип данных - "строительный блок" - оформляется таким образом, что его внутренняя структура становится недоступной для программиста - пользователя этого типа.

Программист, использующий этот тип данных в своей программе (в модуле более высокого уровня), может оперировать с данными этого типа только через вызовы процедур, определенных для этого типа.

Новый тип данных представляется для него в виде "черного ящика" для которого известны входы и выходы, но содержимое - неизвестно и недоступно.

Преимущества инкапсуляции

1) средство преодоления сложности,
2) средство защиты от ошибок.

Первая цель достигается за счет того, что сложность внутренней структуры нового типа данных и алгоритмов выполнения операций над ним исключается из поля зрения программиста-пользователя.

Вторая цель достигается тем, что возможности доступа пользователя ограничиваются лишь заведомо корректными входными точками, следовательно, снижается и вероятность ошибок.

Структуризация данных и ООП

Современные языки программирования блочного типа (Pascal, C) обладают достаточно развитыми возможностями построения программ с ***модульной структурой*** и управления доступом модулей к данным и процедурам.

Расширения же языков дополнительными возможностями конструирования типов и их инкапсуляции делает язык ***объектно-ориентированным***.

Сконструированные и полностью закрытые типы данных представляют собой ***объекты***, а процедуры, работающие с их внутренней структурой - ***методы*** работы с объектами.

При этом в значительной степени меняется и сама концепция программирования. Программист, оперирующий объектами, указывает в программе ЧТО нужно сделать с объектом, а не КАК это надо делать.

Технология программирования и технология баз данных

Причины возникновения некоторых терминологических и понятийных различий в подходе к данным в технологии баз данных и технологии языков программирования:

- развивались параллельно и не всегда согласованно друг с другом.
- объективные различия в природе задач, решаемых системами управления базами данных (СУБД) и системами программирования.

Терминологические и понятийные различия

Ключевым понятием в СУБД является понятие ***модели данных***, в основном тождественное понятию ***логической структуры данных***.

Физическая структура данных в СУБД не рассматривается вообще. Но сами СУБД являются программными пакетами, выполняющими отображение физической структуры в логическую (в модель данных).

Для реализации этих пакетов используются те или иные системы программирования, разработчики СУБД, следовательно, имеют дело со структурами данных в терминах систем программирования.
Для пользователя же внутренняя структура СУБД и физическая структура данных совершенно прозрачна; он имеет дело только с моделью данных и с другими понятиями логического уровня.