**Лекция 11. Типы данных. Структуры данных. Абстрактный тип данных**

**Тип данных** - определяет множество допустимых значений, формат их хранения, размер выделяемых памяти и набор операций, которые можно производить над данными.

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 объемы памяти и обрабатываться с помощью различных команд процессора.

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.

**Простой тип данных** - тип данных, объекты (переменные или константы) которого имеют доступной программисту внутренней структуры. Как правило, к простому относятся числовой, символьный, логический и некоторые другие типы.

**Сложный тип данных**- тип данных, объекты (переменные или константы) которого имеют внутреннюю структуру, доступную программисту. Сложный тип состоит из элементов, относящихся к простым типов. К сложным типам данных относятся: массивы, множества, строки, записи, файлы, динамические переменные, вказивнкы; линейные списки (стеки, очереди), нелинейные списки (двоичные деревья, несимметричные деревья, тексты, графы), процедурный тип, объекты.

**Полиморфическое тип данных** - представление набора типов как единого типа. Есть языки, не связывают изменении, константы, формальные параметры и возвращаемого значения функций с определенными типами, поддерживая единый полиморфный тип данных. В чистом виде таких языков не встречается, но близкие примеры - MS Visual Basic, Delphi - тип variant, Пролог, Лисп - списки. В этих языках переменная может принимать значение любого типа, в параметры функции можно передавать значения любых типов, и вернуть функция также может иметь значение любого типа. Сопоставление типов значений переменных и параметров с применяемыми к ним операциями осуществляется непосредственно при выполнении этих операций. Например, выражение a + b, может трактоваться как сложение чисел, если a и b имеют числовые значения, как Конкатенация строк, если a и b имеют срочные значение, и как недопустимая (ложная) операция, если типы значений a и b несовместимые. Такой порядок называют «динамической типизацией» (соответствует понятию полиморфизм в ООП, полиморфный тип в теории типов). Языки поддерживают только динамическую типизацию, называют иногда «бестиповым». Это название не следует понимать как признак отсутствия понятия типов в языке - типы данных все равно есть.

**Простые типы данных**

**Целый тип** - числа без запятой, могут быть со знаком, то есть принимать как положительные, так и отрицательные значения; и без знака, то есть принимать только положительные значения.

**Действительный тип -** числа с запятой (то есть сохраняются знак и цифры целой и дробной частей) или с плавающей запятой (то есть число приводится к виду m \* be, где m - мантисса, b - пидстанова показательной функции, e - показатель степени (порядок, или экспонента).

**Символьный тип -** сохраняет один символ, в определенном кодировке.

**Логический тип** - может принимать два возможных значения, иногда называются «истиной» и «недостатком» (также «да» и «нет»). При троичной логике может иметь и третье значение - «не определен» (или «неизвестно»).

**Указатель** - сохраняет адрес переменной, на которую ссылается, диапазон значений состоит из адресов ячеек памяти комьютера, нулюва адрес, не является реальной и говорит о том, что в данный момент указатель пустой.

**сложные типы**

**Массив** - индексированный набор элементов одного типа. Одномерный массив - вектор, двумирний массив - матрица.

**Строчный тип** - произвольная последовательность символов алфавита. Каждая переменная такого типа может быть представлена ​​фиксированным количеством байтов или иметь произвольную длину.

**Просчитанный тип** -явно указаны (перечислены) все возможные значения. Преимущества в том, что кроме заданных значений переменных этого типа не смогут принимать больше никаких значений. Кроме того, значение можно задавать вполне осмысленные - например слова. Это упростит понимание кода и написание программы.

**Множество -** конечная совокупность элементов, принадлежащих некоторому базовому типу.

**Запись** - набор различных элементов (полей записи), сохранен как единое целое. Возможен доступ к отдельным полям записи.

**Файловый тип** - сохраняет только однотипные значения, доступ к которым осуществляется только последовательно (файл с произвольным доступом, включенный в некоторые системы программирования, фактически является неявными массивом).

**Абстрактные типы данных**

Абстрактный тип данных (АТД) - это тип данных, предоставляет для работы с элементами этого типа определенный набор функций, а также возможность создавать элементы этого типа с помощью специальных функций. Вся внутренняя структура такого типа спрятана от разработчика программного обеспечения - в этом и состоит суть абстракции. Абстрактный тип данных определяет набор независимых от конкретной реализации типа функций для оперирования его значениями. Конкретные реализации АТД называются структурами данных.

В программировании абстрактные типы данных обычно представляются в виде интерфейсов, которые скрывают соответствующие реализации типов. Программисты работают с абстрактными типами данных исключительно через их интерфейсы, поскольку реализация может в будущем измениться. Такой подход соответствует принципу инкапсуляции в объектно-ориентированный программировании. Сильной стороной этой методики является именно сокрытие реализации. Раз снаружи опубликован только интерфейс, то пока структура данных поддерживает этот интерфейс, все программы, работающие с заданной структурой абстрактным типом данных, будут продолжать работать. Разработчики структур данных стараются,

Не меняя внешнего интерфейса и семантики функций, постепенно дорабатывать реализации, улучшая алгоритмы по скорости, надежности и используемой памяти.

Различие между абстрактными типами данных и структурами данных, которые реализуют абстрактные типы, можно объяснить на следующем примере. Абстрактный тип данных список может быть реализован с помощью массива или линейного списка, с использованием различных техник динамического выделения памяти. Однако каждая реализация определяет тот же набор функций, который должен работать одинаково (по результатам, а не по скорости) для всех реализации.

Абстрактные типы данных позволяют достичь модульности программных продуктов и иметь несколько альтернативных взаимозаменяемых реализаций отдельного модуля.

**Примеры АТД:**

**Список** - множество (постоянная или временная) связанных объектов, упорядоченных некоторым логическим способом. Для обозначения начала списка используется элемент, который называется головой списка.

**Стек -** разновидность линейного списка, структура данных, которая работает по принципу (дисциплиной) «последним пришел - первым ушел» (LIFO, англ. Last in, first out). Все операции (например, удаление элемента) в стеке можно проводить только с одним элементом, который находится на вершине стека и был введен в стек последним.

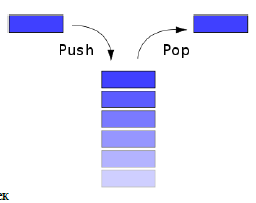


Рисунок 1 - Стек

**Дек** - разновидность линейного списка, в котором элементы могут добавляться как на начало, так и в конце.

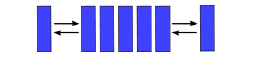


Рисунок 2 - Дек

**Очередь** - динамическая структура данных, работает по принципу "первый пришел - первый ушел» (англ. FIFO - first in, first out). В очереди есть голова (англ. Head) и хвост (англ. Tail). Добавляемый элемент в очередь, оказывается в ее хвосте. Элемент удаляется из очереди, находится в голове.

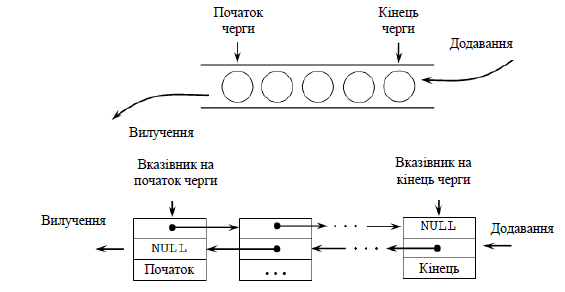


Рисунок 3 - Принцип работы очереди

**Очередь с приоритетом** - это структура данных, которая предназначена для обслуживания множества S, с каждым элементом которой связано определенное значение, называется ключом (англ. Key). Возможны операции: вставка элемента x в множество S, возвращение элемента множества S с наибольшим ключом, возвращение элемента с наибольшим ключом, удаляя его при этом из множества S, изменение значения ключа для элемента x, путем замены его ключом со значением k.

**Ассоциативный массив** (также известный как словарь или карта) -абстрактний тип данных (интерфейс к хранилищу данных), что позволяет хранить данные в виде набора пар ключ - значение и доступ к значениям по их ключом.

**Граф** - совокупность узлов и ребер, соединяющих эти узлы.

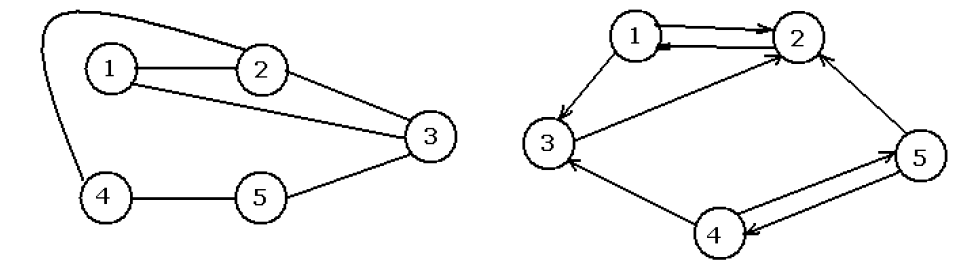


Рисунок 4 - Графы (неориентированный слева, ориентированный справа)

**Дерево** - связан граф, не содержащий циклов, состоит из корневого узла, узлов и листьев.

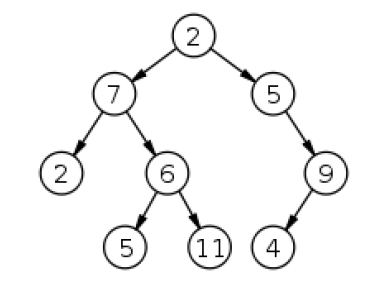


Рисунок 5 - Дерево

**Куча** (поленницу или пирамида) - специализированная древовидная структура данных, в которой существуют определенные свойства упорядоченности: если В - узел-потомок А, тогда ключ (A) ≥ ключ (B). Из этого следует, что элемент с наибольшим ключом всегда является корневым узлом.

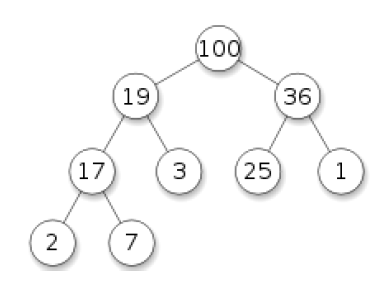


Рисунок 6 - Поленница