Сириус
IT-Колледж

Лекция 5. Структуры данных.

Элементарные структуры данных. Односвязный список, двусвязный список. Абстрактные типы данных, интерфейс и реализация. Стек, очередь, дек, множество, куча; моделирование на основе списка и на основе массива. Массивы переменного размера: аддитивная и мультипликативная схемы аллокации.

Сириус IT-Колледж

Элементарные структуры данных

- **Булевый тип данных**: True (истина) и False (ложь), используется для выполнения логических операций и принятия решений в программе.
- **Указательный тип данных**: содержит адрес памяти, где хранится значение другого типа данных; используется для работы с динамической памятью, аллокации и освобождения памяти, передачи данных между функциями и другими операциями непосредственно с памятью. (отсутствуют в Python)
- **Числовой тип данных**: числовые значения и используются для выполнения математических операций и хранения числовых данных. В различных языках программирования есть разные числовые типы данных, такие как целые числа (integers), числа с плавающей точкой (floating-point), числа с фиксированной точкой (fixed-point) и другие.
- **Символьный тип данных**: представляет отдельные символы или символьные строки (набор символов); используется для хранения и обработки текстовой информации, например, строк текста, обычно включают символы ASCII или Юникода.

Сириус IT-Колледж

Абстрактные типы данных, интерфейс и реализация

Абстрактный тип данных (АТД) — некоторая математическая или информационная модель с совокупностью операторов, определенных в рамках этой модели. Например: множество целых чисел с операторами объединения, пересечения и разности множеств.

Абстрактный тип данных (АТД) — это тип данных (набор значений и совокупность операций для этих значений), доступ к которому осуществляется только через интерфейс.

Интерфейс — заранее определенный набор операций для структуры данных.

Реализация — программа содержащая спецификацию АТД.

Сириус IT-Колледж

Односвязный список

Линейный однонаправленный список — это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

В информатике линейный список обычно определяется как абстрактный тип данных (АТД), формализующий понятие упорядоченной коллекции данных. На практике линейные списки обычно реализуются при помощи массивов и связных списков.

Сириус IT-Колледж

Односвязный список

АТД нетипизированного изменяемого списка может быть определён как набор из конструктора и основных операций:

- Операция, проверяющая список на пустоту.
- Три операции добавления объекта в список (в начало, конец или внутрь после любого (n-го) элемента списка);
- Операция, вычисляющая первый (головной) элемент списка;
- Операция доступа к списку, состоящему из всех элементов исходного списка, кроме первого.

Сириус IT-Колледж

Односвязный список

Характеристики

- **Длина списка**. Количество элементов в списке.
- Списки могут быть **типизированными** и **нетипизированными**. Если список типизирован, то тип его элементов задан, и все его элементы должны иметь типы, совместимые с заданным типом элементов списка.
- Список может быть **сортированным** или **несортированным**.
- В зависимости от реализации может быть возможен **произвольный доступ** к элементам списка.

Сириус IT-Колледж

Двусвязный список

Двусвязный список (двунаправленный связный список) - ссылки в каждом узле указывают на предыдущий и на последующий узел в списке.

Как и односвязный список, двусвязный допускает только последовательный доступ к элементам, но при этом дает возможность перемещения в обе стороны.

В этом списке проще производить удаление и перестановку элементов, так как легко доступны адреса тех элементов списка, указатели которых направлены на изменяемый элемент.

Сириус IT-Колледж

Применение связанных списков

Как применяют связные списки:

- Для построения более сложных структур данных.
- Для реализации файловых систем.
- Для формирования хэш-таблиц.
- Для выделения памяти в динамических структурах данных.

Сириус IT-Колледж

Стек

Стек (stack — стопка) — структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека. Притом первым из стека удаляется элемент, который был помещен туда последним, то есть в стеке реализуется стратегия «последним вошел — первым вышел» (last-in, first-out — LIFO).

Операции стека:

- empty — проверка стека на наличие в нем элементов,
- push (запись в стек) — операция вставки нового элемента,
- pop (снятие со стека) — операция удаления нового элемента.

Сириус IT-Колледж

Стек

Как применяют стеки:

- Для реализации рекурсии.
- Для вычислений постфиксных значений.
- Для временного хранения данных, например истории запросов или изменений.

Сириус IT-Колледж

Очередь

Очередь (queue) — это структура данных, добавление и удаление элементов в которой происходит путём операций push и pop соответственно. Притом первым из очереди удаляется элемент, который был помещен туда первым, то есть в очереди реализуется принцип «первым вошел — первым вышел» (first-in, first-out — FIFO). У очереди имеется голова (head) и хвост (tail). Когда элемент ставится в очередь, он занимает место в её хвосте. Из очереди всегда выводится элемент, который находится в ее голове.

Очередь поддерживает следующие операции:

- empty — проверка очереди на наличие в ней элементов,
- push (запись в очередь) — операция вставки нового элемента,
- pop (снятие с очереди) — операция удаления нового элемента,
- size — операция получения количества элементов в очереди.

Сириус IT-Колледж

Очередь

Как применяют очереди:

- Для реализации очередей, например на доступ к определённому ресурсу.
- Для управления потоками в многопоточных средах.
- Для генерации значений.
- Для создания буферов.

Сириус IT-Колледж

Дек

Дек (deque — double ended queue) — структура данных, представляющая из себя список элементов, в которой добавление новых элементов и удаление существующих производится с обоих концов. Эта структура поддерживает как FIFO, так и LIFO, поэтому на ней можно реализовать как стек, так и очередь. В первом случае нужно использовать только методы головы или хвоста, во втором — методы push и pop двух разных концов. Дек можно воспринимать как двустороннюю очередь.

Сириус IT-Колледж

Дек

Дек имеет следующие операции:

- empty — проверка на наличие элементов,
- pushBack (запись в конец) — операция вставки нового элемента в конец,
- popBack (снятие с конца) — операция удаления конечного элемента,
- pushFront (запись в начало) — операция вставки нового элемента в начало,
- popFront (снятие с начала) — операция удаления начального элемента.

Сириус IT-Колледж

Множество

Множество (set) — тип и структура данных в информатике, которая является реализацией математического объекта множество.

Данные типа множество позволяют хранить ограниченное число значений определенного типа без определённого порядка. Повторение значений, как правило, недопустимо. В общем соответствует концепции математического множества. Для этого типа в языках программирования обычно предусмотрены стандартные операции над множествами.

Сириус IT-Колледж

Куча

Куча — это полное двоичное дерево, удовлетворяющее свойству кучи: если узел A — это родитель узла B, то ключ узла A ≥ ключ узла B.

- Если любой узел всегда больше дочернего узла (узлов), а ключ корневого узла является наибольшим среди всех остальных узлов, это **max-куча**.
- Если любой узел всегда меньше дочернего узла (узлов), а ключ корневого узла является наименьшим среди всех остальных узлов, это **min-куча**.

Сириус IT-Колледж

Массивы переменного размера: аддитивная и мультипликативная схемы аллокации

Аддитивная схема аллокации - при добавлении в массив элемента выходящего за пределы выделенной памяти, объём выделенной памяти увеличивается на некоторую константу. Сложность операции аллокации **O(n^2)**.

Мультипликативная схема аллокации - при добавлении в массив элемента выходящего за пределы выделенной памяти, объём выделенной памяти увеличивается в некоторое фиксированное количество раз. Сложность операции аллокации **O(n)**.