Лабораторная работа №1

Абстрактные структуры данных

Цели и задачи работы: изучение алгоритмов формирования и работы с абстрактными структурами данных.

Задание к работе: Самостоятельно решить задачи в соответствии с индивидуальным вариантом.

Языки программирования: C++ и (Rust или Go). Деревья C++
.

Методика выполнения работы:

1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи (C++, Go или Rust).

3. Протестировать работу программы на различных исходных данных.

4. По запросу преподавателя быть готовым модифицировать алгоритм и добавить операцию работы с данными.

Перечень понятий к защите лабораторной работы 1.

Внимание! Раздел с теорией будет дополнен 2 октября вечером.

Понятие алгоритма. Свойства алгоритма. Алгоритм Евклида. Математический анализ алгоритма. Временная сложность алгоритма. Асимптотическая нотация. Оценки скорости роста функции. Базовые приемы при анализе программы. Основные классы алгоритмов по временной сложности. Графическое преставление скорости роста алгоритмов. Исполнители алгоритмов.

Обоснование выбора алгоритма. Тип данных. Скалярные типы данных. Структурированные типы данных. Фундаментальные структуры. Динамические структуры. Концепция абстрактного типа данных. Понятие структуры данных. Инкапсуляция. Интерфейс. Внутренняя реализация. Интерфейс абстракции. Классификация структур хранения данных. Понятие линейного списка. Последовательность формирования АТД в виде линейного списка. Классификация линейных списков (достоинства и недостатки разных видов линейных списков).

Соглашения при организации связного списка (4 пункта). Эффективность связанного списка (доступ, поиск, вставка, удаление). 4 способа добавления элементов в список. Задача удаления элемента из списка. Вывод элементов списка.

Стек. Эффективность стека на основе связанного списка (добавление, удаление, поиск). Добавление элементов в стек. Задача удаления элемента из стека.

Очередь. Эффективность очереди (добавление, удаление).

Деревья. Определение дерева через АТД и операции. Свойство бинарного дерева. Обозначение элементов дерева. Классификация деревьев. Обходы деревьев в глубину (inorder, postorder, preorder), в ширину. Сравнение обходов деревьев DFS и BFS. Задача поиска элемента в дереве. Задача поиска предшествующего и последующего элементов в дереве. Вставка элемента. Удаление

элемента (виды, схема, псевдокод).

Бинарное дерево. Бинарное дерево поиска. Виды обхода деревьев. Full Binary Tree. Complete Binary Tree. Сбалансированное двоичное дерево (АВЛ-дерево). Эффективность деревьев (через О-нотацию). Красно-черные деревья.

Хеш-таблица. Эффективность хеш-таблицы. Виды коллизий и алгоритмы по их устранению.

Ссылки на сайты с визуализациями:

https://clck.ru/35opVK

https://clck.ru/35opWV,

https://clck.ru/LS6WD

Реализация абстрактных структур данных.

1. Реализовать структуры данных с базовым набором операций:
    a. Массив. Операции: создание, добавление элемента (по индексу и в конец массива), получение элемента по индексу, удаление элемента по индексу, замена элемента по индексу, длина массива, чтение.
    b. Односвязный список. Операции: добавление и удаление элемента (4 способа (до после голова хвост)), чтение (разное, несколько способов), удаление элемента по значению, поиск элемента по значению.
    c. Двусвязный список. Операции: добавление и удаление элемента (4 способа (до после голова хвост)), чтение (разное, несколько способов), удаление элемента по значению, поиск элемента по значению.
    d. Стек. Операции: добавление и удаление элемента (push и pop), чтение.
    e. Очередь. Операции: добавление и удаление элемента (push и pop), чтение.
    f. Деревья (вариант 1[1] Красно-черное дерево, вариант 2 - Сбалансированное двоичное дерево (АВЛ-дерево), вариант 3 Complete Binary Tree, вариант 4 Full Binary Tree, вариант 5 – бинарное дерево поиска. Операции: добавление элемента, поиск элемента, удаление элемента с сохранением структуры дерева, чтение. Если Full Binary Tree или Complete Binary Tree, то вместо удаления элемента добавить операцию проверки на Full или Complete соответственно.
2. Реализовать интерфейс работы со структурами данных. Предусмотреть считывание из файла и запись в файл при внесении изменений в состав элементов.
   Часть операций для реализации представлено в таблице. Дополнить список операций согласно п.1 и сохраняя стиль написания команды (первая буква – принадлежность структуре данных). **Исключение: Операция PRINT выводит любую структуру данных на экран.**

Примерная таблица операций

Тип контейнера	Добавление	Удаление	Чтение
Массив (M)	MPUSH	MDEL	MGET
Односвязный список (F)	FPUSH	FDEL	FGET
Двусвязный список (L)	LPUSH	LDEL	LGET
Очередь (Q)	QPUSH	QPOP	QPOP
Стек (S)	SPUSH	SPOP	SPOP
Дерево (T)	TINSERT	TDEL	TGET

[1] Вариант 1 – семестр сдан на 5, Вариант 2 – семестр сдан на 4, Вариант 3 – семестр сдан на 3, Вариант 4 – Семестр не сдан, на 26.09.2025 сдано 80% работ, Вариант 5 – остальные.

3. Обосновать сложность выполнения каждой операции с позиции BigO нотации. Составить таблицу BigO по операциям и структурам в отчете.
4. Представить достоинства и недостатки каждой структуры данных, описать сферу применения (примеры).
5. В отчете в контрольном примере представить визуальные формы, подтверждающие соответствие результатам работы программы.

Пример

```
./dbms --file file.data --query 'HSET myhash key value'
```

Разбор примера

- ./dbms имя нашей программы, используя ./ запускаем программу
- --file file.data используя ключ --file указываем файл в котором содержаться данные с которыми будет работать СУБД
- --query 'HSET myhash key value' используя ключ --query делаем запрос к базе данных, текст означает - положить в хэш-таблицу myhash ключ key со значением value

Дополнительные примеры

Добавление элемента item в стек с именем mystack

```
./dbms --file file.data --query 'SPUSH mystack item'
-> item
```

Чтение и удаление элемента из очереди с именем myqueue

```
./dbms --file file.data --query 'QPOP myqueue'
-> element
```

Проверка содержится ли элемент value в дереве с именем mytree

```
./dbms --file file.data --query 'ISMEMBER mytree value'
-> TRUE
```

Литература

1. Т. Кормен. Алгоритмы. Построение и анализ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн . – Издательство «Вильямс», 2013.

2. Дж. Макконелл Анализ алгоритмов. Активный обучающий подход. — 3-е дополненное издание. М: Техносфера, 2009. -416с.

3. Миллер, Р. Последовательные и параллельные алгоритмы: Общий подход / Р. Миллер, Л. Боксер ; пер. с англ. — М. : БИНОМ. Лаборатория знаний, 2006. — 406 с.

4. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. — СПб.: БХВ-Петербург. 2011. — 720 с.: ил.

5. Фундаментальные алгоритмы на C++ часть 1-4 (Роберт Седжвик) 2001 (или другие книги по алгоритмам данного автора).