Отчет по теме №1 АДТ. Коллекции

Работу выполнил:

Студент группы ИВТ(ВМК)-21

Рычков Родион Викторович

Абстрактный тип данных (ADT) — это математический абстрактный тип, который определяет набор операций и свойств, но не определяет способ их реализации. В программировании ADT используется для определения интерфейса объекта, который скрывает его реализацию и позволяет работать с ним только через определенные методы и операции.

Коллекции - это ADT, предназначенный для хранения и управления группами объектов. Они являются основным инструментом для работы с данными в большинстве программных языков. Коллекции бывают разных типов, некоторые из них представлены ниже:

1. Списки Списки позволяют хранить элементы в определенном порядке, присваивая каждому элементу индекс. Операции над списками могут включать добавление, удаление, поиск, вставку и сортировку элементов. В языке Python, например, списки могут быть реализованы с помощью встроенного типа данных "list".

2. Множества Множества представляют собой коллекции уникальных элементов, неупорядоченных между собой. Операции над множествами могут включать добавление, удаление, проверку на наличие элемента и выполнение математических операций над множествами (объединение, пересечение и разность). В языке Python, например, множества могут быть реализованы с помощью встроенного типа данных "set".

3. Отображения (словари) Отображения (или словари) позволяют хранить пары ключ-значение и обеспечивают быстрый доступ к значениям по ключу. Операции над отображениями могут включать добавление, удаление, поиск и изменение элементов. В языке Python, например, словари могут быть реализованы с помощью встроенного типа данных "dict".

4. Стеки Стеки представляют собой коллекции элементов, доступ к которым осуществляется только через одну конечную точку. Операции над стеками могут включать добавление элемента (называемого "помещением" или "пушем" на стек), удаление элемента (называемого "извлечением" или "попом" со стека) и проверку наличия элементов. В языке Python, например, стеки могут быть реализованы с помощью модуля "collections" и методов "append" и "pop".

5. Очереди Очереди представляют собой коллекции элементов, доступ к которым осуществляется по принципу "первый вошел - первый вышел" (FIFO). Операции над очередями могут включать добавление элемента в конец очереди (называемого "вступлением" или "энкью"), удаление элемента из начала очереди (называемого "выбыванием" или "декью") и проверку наличия элементов. В языке Python, например, очереди могут быть реализованы с помощью модуля "collections" и методов "append" и "popleft".

6. Деревья Деревья представляют собой коллекции элементов, которые имеют иерархическую структуру. Один элемент является корневым, а остальные элементы могут иметь родительский элемент и несколько дочерних элементов. Операции над деревьями могут включать добавление элемента, удаление элемента, поиск элемента и обход элементов в заданном порядке. В языке Python, например, деревья могут быть реализованы с помощью модуля "binarytree".

7. Графы Графы представляют собой коллекции элементов, которые имеют связи между собой. Элементы называются вершинами, а связи между ними - ребрами. Операции над графами могут включать добавление вершины, добавление ребра, удаление вершины, удаление ребра и обход вершин в заданном порядке. В языке Python, например, графы могут быть реализованы с помощью модуля "networkx".

Коллекции имеют различные свойства и характеристики, такие как производительность, порядок элементов, уникальность элементов и другие. Выбор коллекции для конкретной задачи зависит от требований к производительности и функциональности, а также от особенностей используемого языка программирования.

8. Словари Словари представляют собой коллекции пар "ключ-значение". Операции над словарями могут включать добавление пары ключ-значение, удаление пары ключ-значение, получение значения по ключу и проверку наличия ключа. В языке Python, например, словари могут быть реализованы с помощью встроенной структуры данных "dict".

9. Множества Множества представляют собой коллекции уникальных элементов без упорядоченности. Операции над множествами могут включать добавление элемента, удаление элемента и проверку наличия элемента. В языке Python, например, множества могут быть реализованы с помощью встроенной структуры данных "set".

10. Стеки Стеки представляют собой коллекции элементов, доступ к которым осуществляется по принципу "последний вошел - первый вышел" (LIFO). Операции над стеками могут включать добавление элемента на вершину стека (называемое "помещением" или "пушем"), удаление элемента с вершины стека (называемого "извлечением" или "попом") и проверку наличия элементов. В языке Python, например, стеки могут быть реализованы с помощью встроенной структуры данных "list".

11. Очереди с приоритетом Очереди с приоритетом представляют собой коллекции элементов, которые упорядочены по заданному приоритету. Операции над очередями с приоритетом могут включать добавление элемента с определенным приоритетом, удаление элемента с наивысшим приоритетом и проверку наличия элементов. В языке Python, например, очереди с приоритетом могут быть реализованы с помощью модуля "heapq".

Коллекции играют важную роль в программировании и предоставляют различные инструменты для работы с данными. Знание различных типов коллекций и их характеристик помогает выбрать наиболее подходящую коллекцию для конкретной задачи и повысить производительность программы.

**Источники:**

[**https://russianblogs.com/article/45822219093/**](https://russianblogs.com/article/45822219093/)

[**https://kzen.dev/ru/74795809**](https://kzen.dev/ru/74795809)

**https://themoney.co/ru/what-does-adt-stand-for-in-programming/**