**Расписать в текстовом документе плюсы и минусы изученных структур данных и расписать примеры, в которых нужно будет использовать тот или иной контейнер.**

**ArrayList**

Описание: динамический массив, для хранения вместе разнотипных объектов - строк, чисел и т.д.

**-** нет типовой безопасности

**+** манипуляция объектом любого типа, емкость увеличивается автоматически, доступ к любому элементу

**List<T>**

Описание: динамический массив, функционал как у класса ArrayList.

**-** медленный поиск, удаление и вставка, сортировка не гарантируется, необходимо отсортировать перед выполнением операций

**+** типобезопасный,емкость увеличивается автоматически, быстрый доступ к любому элементу по индексу, быстрое добавление в начало и конец, может быть null элемент, элементы могут дублироваться

**LinkedList<T>**

Описание: сохраняет элементы в двунаправленном списке, каждый элемент хранит ссылку на следующий и предыдущий элементы.

**-** все элементы связных списков доступны лишь друг за другом, нужна дополнительная память под ссылки

**+** типобезопасный,операция вставки элемента в середину выполняется быстро, может быть null элемент, элементы могут дублироваться

Изображение выглядит как стол

Автоматически созданное описание

**HashSet<T>**

Описание: коллекция, содержащая только отличающиеся элементы, сохраняет уникальные значения.

**-** элементы не могут дублироваться, не сортируется

**+** может увеличить производительность, часть операций выполняется быстрее, емкость увеличивается автоматически, можно производить операции над множествами, linq

**SortedSet<T>**

Описание: отсортированное множество (список отличающихся элементов).

**-** элементы не могут дублироваться, не может менять значения сортировки

**+** может увеличить производительность, часть операций выполняется быстрее, емкость увеличивается автоматически, можно производить операции над множествами

**ObservableCollection<T>**

Описание: функционал похож на List<T>, динамическая коллекция данных, выдает уведомления при добавлении, удалении элементов и при обновлении списка.

**ReadOnlyCollection<T>**

Описание: базовая коллекция, с оболочкой, которая предотвращает внесение изменений, доступна только для чтения.

**KeyValuePair<K, V>**

Описание: служит для хранения ключа и его значения, применяется в классах обобщенных коллекций, например в Dictionary

**Dictionary<K, V>**

Описание: сохраняет пары "ключ-значение”, доступ к значению осуществляется по ключу

**-** сложная сортировка значений, нельзя менять ключ (должен быть уникальным, не null)

**+** динамический, но можно зафиксировать значение, удобный доступ к элементам

**SortedDictionary<T>**

Описание: отсортированный список из пар "ключ-значение", упорядочен по ключу

**-** нельзя менять ключ, нельзя изменить значения сортировки

**+** более быстрые операции вставки и удаления для несортированных данных, удобный доступ к элементам

**Queue<T>**

Описание: коллекция, в которой элементы обрабатываются по схеме "первый вошел, первый вышел" (FIFO), элемент, вставленный в очередь первым, первым же и читается. Используется, когда может потребоваться отменить элемент после получения его значения. Емкость увеличивается автоматически, принимает null, допускает дублирование.

**Stack<T>**

Описание: коллекция переменного размера экземпляров одинакового типа, обслуживается по принципу "последним пришел - первым вышел" (LIFO). . Используется, когда может потребоваться отменить элемент после получения его значения. Используется, когда может потребоваться отменить элемент после получения его значения. Емкость увеличивается автоматически, принимает null, допускает дублирование.