**Задача.** List

**Лекции, нужные для решения задачи:** 1-9.

1. Сделать классы для односвязного списка и узла списка.

Для эффективности сделайте поле для хранения длины списка.

Надо реализовать методы:

* получение размера списка
* получение значение первого элемента
* получение/изменение значения по указанному индексу.   
  Изменение значения по индексу пусть выдает старое значение.
* удаление элемента по индексу, пусть выдает значение элемента
* вставка элемента в начало
* вставка элемента по индексу
* удаление узла по значению, пусть выдает true, если элемент был удален
* удаление первого элемента, пусть выдает значение элемента
* разворот списка за линейное время
* копирование списка

2\* (Эта задача просто для ознакомления, проверяться она не будет). Есть односвязный список, каждый элемент которого хранит дополнительную ссылку на произвольный элемент списка. Эта ссылка может быть и null.

Надо создать копию этого списка, чтобы в копии эти произвольные ссылки ссылались на соответствующие элементы в копии.

Чему научитесь:

* Понимание односвязных списков
* Это достаточно сложно алгоритмически
* Generic’и

**Задача.** ArrayList

**Лекции, нужные для решения задачи:** 1-8, 10 понятие итератора, 13.

Сделать свою реализацию списка ArrayList<T> (в C# - List<T>), сделать его generic’ом.

Необходимо реализовать интерфейс List<T> (в C# - IList<T>).

Нужно реализовать специфичный конструктор, принимающий вместимость, а также методы ensureCapacity и trimToSize (в C# - свойство Capacity и метод TrimExcess).

В Java методы sublist и listIterator реализовывать не нужно.

Чему научитесь:

* Понимание работы списка на массиве
* Реализация сложного интерфейса
* Реализация итератора
* Больше будете думать о временных оценках
* Generic’и

**Задача**. HashTable

**Лекции, нужные для решения задачи:** 1-8, 10 понятие итератора, 13.

Сделать свою реализацию хэш-таблицы, сделать ее generic’ом.

Класс должен реализовать интерфейс Collection<T> (ICollection<T> в C#).

И в одном из конструкторов сделать параметр, который задает размер массива хэш-таблицы.

Чему научитесь:

* Понимание работы хэш-таблицы
* Реализация интересного итератора
* Generic’и

**Задача \*.** Tree

**Лекции, нужные для решения задачи:** 1-8, 14.

Реализовать бинарное дерево поиска.

Нужны операции:

1. Вставка
2. Поиск узла
3. Удаление первого вхождения узла по значению
4. Получение числа элементов
5. Обходы в ширину и глубину. Обход в глубину двумя вариантами – с рекурсией и без

Чему научитесь:

* Понимание работы с деревьями
* Обходы в глубину и ширину

**Задача \*.** Graph

**Лекции, нужные для решения задачи:** 1-8, 14.

Реализовать обход несвязного графа в ширину и глубину.

Граф просто задаем двумерным массивом.

Чтобы реализовать обходы для несвязного графа надо просто посмотреть массив visited после обхода очередной компоненты связности.

Если остались не visited вершины, запускаем алгоритм для них и т.д., пока такие вершины не кончатся.

Чему научитесь:

* Понимание как работать с графами