**Первый слайд:**

Здравствуйте. Меня зовут Оганесян Левон, тема моей дипломной работы Разработка библиотеки генерации тестов для задач по информатике. Научный руководитель: Армен Цирунян.

**Второй слайд:**

Димломная работа - это библиотека для генерации тестов, написанная на языке C++. При разработке использован не один шаблон проектирования, алгоритмы, реализована работа в нескольких потоках.

Библиотека предоставляет такие инструменты, как диапазон значений, массивы на диапазонах, случайные графы, регулярные выражения, грамматики и многое другое.

**Третий слайд:**

Что же такое **диапазон значений**? При формулировке олимпиадных задачах все переменные во входных данных имеют какой-то допустимый диапазон – это значит, что данные переменные гарантированно будут в данных пределах. В рамках данной работы будем называть этот диапазон **диапазоном значений.** Все созданные данной библиотекой переменные и тесты должны не выходить за рамки соответствующих им диапазонов. Вот примеры **диапазонов значений:** число, размер массива, количество графов.

**Четвертый слайд:**

Для понимания, что же должна уметь делать данная библиотека приведем в пример несколько типичных задач: В данной задаче нужно сгенерировать число n, которое лежит от 1 до 100000, число m, которое лежит от n до 100000, и так же n штук пар ai иbi, которые лежат от 1 до 1000. Данная задача продемонтирует, как устроены диапазоны, массивы и зависимые переменные в библиотеке.

**Пятый слайд:**

Данная задача требует определить является ли данный граф лесом. Напомню, лес это множество деревьев. В данной задаче нужно сгенерировать 2 типа тестов: лес и простой случайный граф. Для первых ответ должен быть Yes, для оставшейся части No. Дано количество вершин, дано количество ребер, и далее перечислены пары чисел – ребра в нашем графе. Данная задача продемонтрирует класс графов.

**Шестой слайд:**

Данная задача на ввод даёт строку, которая содержит в себе ровно один символ «@». С обоих сторон данного знака непустые строки состоящие из символов английского алфавита и точки. При этом две точки не могут находиться рядом. Данная задача продемонстрирует класс регулярных выражений.

**Седьмой слайд:**

Так выглядит архитектура моей библиотеки. Test - основной класс, от которого наследуются все остальные классы. Как можно заметить, есть классы Array, CompositeTest, ConstStringSet, Delimiter, Graph, GraphMerger, Matrix, PrimitiveTest и другие, о каждом сейчас будет подробно рассказано.

**Восьмой слайд:**

Начнем с описания класса Test. Это базовый абстрактный класс, который содержит весь необходимый функционал для работы с тестами. Является частью шаблона проектирования Компоновшик.

**Девятый слайд:**

CompositeTest – так же один из основных классов шаблона Компоновщик. Содержит в себе множество Тестов, которыми можно управлять как одним. Для добавления в множество нужно использовать функцию Add.

**Десятый слайд:**

PrimitiveTest – класс описывающий примитивные типы типа int, double, char в рамках данной библиотеки. Зачем это нужно? Напомню, что мы должны работать не просто с числами, а с **диапазонами,** поэтому нам для этого нужен свой собственный тип. PrimitiveTest разделяется на ConstPrimitiveTest, который описывает константу и RangePrimitiveTest, который описывает **диапазоны**.

**Одиннадцатый слайд:**

Опеределим, что такое зависимые и независимые тесты. Зависимые тесты – это те тесты, которые каким-то образом зависят от других тестов. Вот пример зависимого теста: создаём элемент от 1 до 1000, а потом его же используем для создания следующего элемента. Все остальные тесты назовем независимыми. Это разделение введено, так как во время разработки оказалось, что архитектура приложения не может нормально обрабатывать зависимые тесты, и стало нужно писать методы работающие именно с зависимыми тестами.

**Двенадцатый слайд:**

В контексте нашей библиотеки нам так же нужны массивы, которые работают с Тестами. Массивы, разумеется, умеют работать, как с зависимыми так и с независимыми тестами. Так же присутствует класс матриц, который имеет тот же интерфейс, что и массив.

**Тринадцатый слайд:**

Класс грамматики представляет из себя контекстно-свободную граматику. Принимает 2 строки, первая – описание грамматики, вторая – правила. Пример данного класса можно увидеть в документе.

**Четырнадцатый слайд:**

Класс представляющий регулярные выражения. Данный класс генерирует строки, которые подходят данному регулярному выражению. Конструктор класса принимает одну строку и одно необязательное целое число – максимальную глубину для операций \* и +, которая по умолчанию является 1000. Строка же является именно регулярным выражением, по которому и нужно генерировать строки. Задача 3 будет решена с помощью этого класса.

**Пятнадцатый слайд:**

Класс Graph. Данный класс генерирует случайные графы, на данный момент поддерживаются ориентированные и неориентированные, взвешенные графы, с определеным методом вывода. На данный момент методов вывода три: матрица смежности, список смежности и список ребер. Матрица смежности это такая матрица размера nxn (n это количество вершин), где ячейка ij равна единице, если есть ребро из вершины i ы вершину j и 0 в обратном случае. Список смежности это список, где i-ая ячейка показывает какие соседи есть у i-той вершины. Список ребер это просто список всех ребер графа, перечисленных друг за другом как пары. При этом только при списке ребер можно иметь веса.

Конструктор класса принимает количество вершин, количество ребер и веса, если они есть.

**Шестнадцатый слайд:**

Генерация графа была одна из самых сложных задач данной библиотеки, поэтому нужно описать этот процесс. Итак, изначально фиксируется произвольное дерево. Далее определяется граф большой или маленький. Граф маленький если в нем должно быть меньше половины всех ребер и большой в обратном случае. Если граф маленький, то граф начинает строиться начиная с дерева. Если же граф большой, то создается полный граф и из него начинают удаляться ребра, и при этом не удаляя ребра фиксированного дерева. Разделение на маленький и большой граф способствует ускорению построения графа. Отсюда следует, что граф всегда связный. Но что если нам нужно сгенерировать несвязный граф?

**Семнадцатый слайд:**

Класс GraphMerger создан с целью объединять графы. Он принимает массив графов. Алгоритм объединения таков: узнаётся общее количество вершин (предположим, n), создается массив {1, 2, 3, …, n} и случайным образом данный массив перемешивается. Каждой вершине присваивается свой новый номер, создается лист смежности размера n и заполняется листами смежности каждого отдельного графа. В результате получается большой граф, который является объединением всех переданных графов, при этом номера вершины перемешаны.

**Восемнадцатый слайд:**

Мы можем сгенерировать тест и вывести его в стандарный выходной поток или в созданный нами выходной поток, но что если мы хотим сгенерировать много тестов и записать их в разные файлы с определенным расширением и с порядковым номером? Это было бы неудобно делать для каждой задачи вручную, поэтому был написан этот класс. Основные аргументы данного класса: собственно сам тест, который нужно генерировать, количество тестов для генерации, путь, куда должны эти тесты записаться и какое количество потоков должно быть использовано для генерации.

**Девятнадцатый слайд:**

Теперь пройдём к решениям самих задач нашими методами. Напомню, в этой задаче надо было сгенерировать числа n и m, затем сгенеровать n пар чисел ai bi. В данной библиотеке основной идеей написания теста является парадигма разделяй и властвуй, это мы тут и будем использовать: дробить тест на части. Так например, можно представить, что пара это отдельный тест, массив пар это другой тест, а все вместе третий тест.

**Двадцатый слайд:**

Прочитаем код не в хронологическом порядке. Создаём пару, как отдельный тест. Создаем переменную n и создаем массив состоящий из n пар. Далее создем m, который больше n и объединяем всё.

**Двадцать первый слайд:**

Здесь надо сгенерировать деревья, чтобы всего вершин и ребер было не более 100000 и вывести в виде пар ребер.

**Двадцать второй слайд:**

Не буду вдаваться в подробности, создаётся дерево, добавляется в мерджер, мерджер выводит список пар.

**Двадцать третий слайд:**

Здесь нужно сгенерировать адрес электронной почты, в котором есть один символ “@”, точки и строчные латинские буквы, при этом строка не может начинаться и заканчиваться на точку и иметь две подряд точки.

**Двадцать четвертый слайд:**

Данное регулярное выражение описывает те условая, которые описываются в задаче. Число 10000 говорит какая максимальная грубина может быть для операций + и \*.

**Двадцать пятый слайд:**

16 сентября 2017 года была успешно проведена олимпиада, тесты которой были полностью сгенерированы данной библиотекой. В дальшейшем такого рода олимпиады так же будут организовываться.

**Двадцать шестой слайд:**

Исходные коды можно найти на сайте гитхаб в открытом доступе. Всем спасибо за внимание.