

**Министерство образования и науки Российской Федерации**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ  
ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

**«Синтез исправлений для неверных решений олимпиадных задач  
по программированию»**

Автор: Шовкопляс Григорий Филиппович \_\_\_\_\_

Направление подготовки (специальность): 01.03.02 Прикладная математика и  
информатика

Квалификация: Бакалавр

Руководитель: Буздалов М.В., канд. техн. наук \_\_\_\_\_

**К защите допустить**

Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. \_\_\_\_\_

« \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

Санкт-Петербург, 2017 г.

**Студент** Шовкопляс Г.Ф. **Группа** М3439 **Кафедра** компьютерных технологий  
**Факультет** информационных технологий и программирования

**Направленность (профиль), специализация** Математические модели и алгоритмы  
разработки программного обеспечения

Квалификационная работа выполнена с оценкой \_\_\_\_\_

Дата защиты \_\_\_\_\_ « \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

Секретарь ГЭК \_\_\_\_\_ Принято: « \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

Листов хранения \_\_\_\_\_

Демонстрационных материалов/Чертежей хранения \_\_\_\_\_

**Министерство образования и науки Российской Федерации**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ  
ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»**

**УТВЕРЖДАЮ**

Зав. каф. компьютерных технологий

докт. техн. наук, проф.

\_\_\_\_\_ Васильев В.Н.

«\_\_\_» \_\_\_\_\_ 20\_\_\_ г.

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**Студент** Шовкопляс Г.Ф. **Группа** М3439 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования  
**Руководитель** Буздалов Максим Викторович, канд. техн. наук, доцент кафедры КТ Университета ИТМО

**1 Наименование темы:** Синтез исправлений для неверных решений олимпиадных задач по программированию

**Направление подготовки (специальность):** 01.03.02 Прикладная математика и информатика

**Направленность (профиль):** Математические модели и алгоритмы разработки программного обеспечения

**Квалификация:** Бакалавр

**2 Срок сдачи студентом законченной работы:** «31» мая 2017 г.

**3 Техническое задание и исходные данные к работе.**

Требуется разработать способ анализа кода программ для решения олимпиадных задач, с целью применения автоматического исправления ошибок для повышения продуктивности обучения школьников.

**4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)**

Пояснительная записка должна демонстрировать актуальность данной задачи и подход к решению. Должна быть проведена оценка эффективности, а также оценены перспективы развития.

**5 Перечень графического материала (с указанием обязательного материала)**

Не предусмотрено

**6 Исходные материалы и пособия**

- а) ANTLR 4 Documentation;
- б) Томас Кормен и др. Алгоритмы: построение и анализ;
- в) Earl T. Barr, Mark Harman и др. Automated Software Transplantation.

### 7 Календарный план

№№ пп.	Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Ознакомление с областью задачи, поиск существующих решений	10.2017	
2	Разработка алгоритма для решения задачи	11.2017	
3	Реализация алгоритма	02.2017	
4	Тестирование и оценка эффективности	04.2017	
5	Написание пояснительной записки	05.2017	

**8 Дата выдачи задания:** «01» сентября 2017 г.

Руководитель \_\_\_\_\_

Задание принял к исполнению \_\_\_\_\_ «01» сентября 2017 г.

**Министерство образования и науки Российской Федерации**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ  
ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»**

**АННОТАЦИЯ  
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

**Студент:** Шовкопляс Григорий Филиппович

**Наименование темы работы:** Синтез исправлений для неверных решений олимпиадных задач по программированию

**Наименование организации, где выполнена работа:** Университет ИТМО

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

**1 Цель исследования:** Исследование возможности синтеза исправлений для неверных решений олимпиадных задач по программированию.

**2 Задачи, решаемые в работе:**

- а) Провести анализ актуальности;
- б) Разработать алгоритм для решения;
- в) Оценить эффективность алгоритма и возможности его улучшения.

**3 Число источников, использованных при составлении обзора:** \_\_\_\_\_

**4 Полное число источников, использованных в работе:** 0

**5 В том числе источников по годам**

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет

**6 Использование информационных ресурсов Internet:** \_\_\_\_\_

**7 Использование современных пакетов компьютерных программ и технологий:**

Были использованы языки программирования Java 1.8, Python 3 и генератор синтаксических анализаторов ANTLR. Для Java была добавлена библиотека StructureGraphic для визуализации деревьев, а также библиотека для работы с ANTLR. Для разработки кода использовалась среда IntelliJ IDEA с плагином для запуска ANTLR для написания кода на Java и FAR manager для написания кода на Python.

**8 Краткая характеристика полученных результатов:** Реализованный метод синтеза покрывает достаточно большое количество типичных ошибок, а при дальнейшей доработке способен покрывать абсолютное большинство.

**9 Гранты, полученные при выполнении работы:**

**10 Наличие публикаций и выступлений на конференциях по теме работы:**

Выпускник: Шовкопляс Г.Ф. \_\_\_\_\_

Руководитель: Буздалов М.В. \_\_\_\_\_

« \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	5
1. Обзор .....	6
1.1. Термины и понятия.....	6
1.1.1. Олимпиадное программирование .....	6
1.1.2. Теория графов .....	6
1.1.3. Синтаксический анализ.....	7
1.1.4. Классификация ошибок.....	8
1.2. Code transplantation .....	8
1.3. На будущее. вторая глава вообще? .....	8
1.3.1. Ошибки .....	8
Выводы по главе 1.....	9

## ВВЕДЕНИЕ

В настоящее время популярны и продолжают набирать популярность различные олимпиады по программированию. Особенностью данных олимпиад является использование автоматической системы тестирования, которая проверяет программу участника на заранее подготовленных тестах.

Ведут работу и крайне востребованы различные кружки, онлайн-курсы и другие обучающие занятия в данной области. Преподаватели постоянно ищут ошибки в программах учащихся в процессе обучения. Не всегда найти ошибку просто, особенно небольшую.

При этом одну и ту же задачу решает множество различных школьников, в разные моменты времени, но все послышки хранятся на сервере с тестирующей системой в данном случае, в отличие от, например, математических задач, которые чаще всего проверяются устно, либо письменно, но на бумаге, которую не хранят.

В любой области обучения можно столкнуться с ошибками, которые совершают множество людей при обучении, в математике, например, это потеря корней при раскрытии модуля или деления на зависимую переменную. Программирование не является исключением, и ошибки при решении задач тоже повторяются, однако, как описано выше в рамках одной тестирующей системы хранится архив всех попыток сдачи. Данную информацию как раз и хочется использовать, для повышения эффективности преподавания.

В данной статье будет рассмотрен способ синтеза исправлений для решения учащегося. Это поможет в некоторой степени автоматизировать процесс обучения, а следовательно снять не самую полезную нагрузку с преподавателя. Ведь, если в случае с локальным кружком на одного преподавателя приходится не более двадцати учащихся, существуют онлайн-курсы, в которых количество учащихся в разы больше.



## ГЛАВА 1. ОБЗОР

### 1.1. Термины и понятия

В данном разделе описаны основные термины и понятия, используемые в представленной работе.

#### 1.1.1. Олимпиадное программирование

**Задача** олимпиадного программирования представляет из себя некоторое задание, которое требуется выполнить, написав программу на одном из языков программирования. Чтобы проверить корректность выполнения задания, текст программы отправляют на проверку в тестирующую систему.

Под **тестирующей системой** в данной работе будем подразумевать сервер, на который учащиеся отправляют свой код, для проверки его корректного выполнения на заранее подготовленных тестах.

**Тест** для олимпиадной задачи по программированию — некоторые входные данные, удовлетворяющие условию задачи. Как правило, представляет из себя текстовый файл. Для каждой задачи обычно существует свой набор тестов. В наборе может быть, как один, так и несколько тестов. Тесты составляются до проведения соревнования или занятия и обычно неизвестны учащемуся.

**Решением** олимпиадной задачи называют программу, которая считывает входные данные, находит ответ и выводит его. Также нередко в условии задачи присутствуют ограничения на время выполнения и количество используемой памяти, поэтому корректное решение должно укладываться в эти ограничения.

**Вердикт** тестирующей системы — ответ тестирующей системы после проверки некоторого решения. Может быть положительным или одним из отрицательных. В случае получения положительного ответа, считается, что задача решена правильно, иначе, что в решении присутствует ошибка. Как правило, вместе с вердиктом учащийся получает комментарий с номером теста, на котором решение работает некорректно.

#### 1.1.2. Теория графов

**Граф** — абстрактный математический объект, который характеризует пара  $G = (V, E)$ , где  $V$  — множество вершин, а  $E \subset \{(v, u) : v, u \in V\}$  — множество ребер.

**Связный граф** — граф, в котором между любой парой вершин существует хотя бы один путь.

**Цикл** — последовательность вершин вида  $v_1, v_2 \dots v_k$ , где  $v_i \in V$ ,  $(v_i, v_{i+1}) \in E$  и  $v_1 = v_k$ .

**Ациклический граф** — граф, который не содержит в себе циклов.

**Дерево** — связный ациклический граф.

**Поиск в глубину** — один из наиболее популярных алгоритмов обхода графа, используемый для изучения строения.

### 1.1.3. Синтаксический анализ

**Абстрактное синтаксическое дерево**(дерево разбора) — структура данных, представляющая из себя ориентированное дерево, в котором каждая вершина сопоставляется с оператором языка программирования, а листья с соответствующими операндами.

**Лексический анализ** — процесс аналитического разбора входного текста на известные группы, с последующем получением на выходе идентифицированных последовательностей, называемых «токенами». Лексический анализ используется в компиляторах и интерпретаторах исходного кода языков программирования, и в различных парсерах слов естественных языков.

**Лексическим анализатором**(жарг. лексер от англ. *lexer*) называется программа, выполняющая лексический анализ текста.

**Синтаксический анализ**(жарг. парсинг от англ. *parsing*) — процесс сопоставления последовательности токенов(слов) формального языка с его формальной грамматикой. Результатом будет абстрактное синтаксическое дерево. Как правило, для получения токенов проводится лексический анализ.

**Синтаксическим анализатором**(жарг. парсер от англ. *parser*) называется программа выполняющая синтаксический анализ.

**Генератор синтаксических анализаторов** — программа, которая получает на вход контекстно-свободную грамматику некоторого языка, а на выход выдает синтезированный код лексического и синтаксического анализаторов для данного языка. Для большинства используемых в олимпиадном программировании языков программирования существуют стандартные уже реализованные грамматики.

### 1.1.4. Классификация ошибок

**Ошибкой** будем называть причину, по которой решение получает отрицательный вердикт на одном из тестов.

Множество ошибок можно интуитивно разделить на несколько категорий:

- а) **Идейные.** Данные ошибки совершены по причине написания неправильной интерпритации условия, либо некорректного выбора алгоритма для решения. Простой пример — решение задачи динамического программирования о рюкзаке, используя метод жадного программирования.
- б) **Нарушающие ограничения** ошибки совершаются, потому что решение либо потребляет памяти больше, чем указанное допустимое значение в условии, либо не успевает завершиться раньше, указанного в условии максимального времени работы.
- в) **Неаккуратная реализация**
- г) **Нерассмотренные случаи**
- д) **Мелкие**

## 1.2. Code transplantation

Тут расскажу про подход и почему он не подходит.

## 1.3. На будущее. вторая глава вообще?

### 1.3.1. Ошибки

Рассмотрим случайную программу с ошибкой и поймем, почему нужно рассматривать именно «мелкие» ошибки:

- Ошибки, не являющиеся «мелкими», легко найти, потому что обычно они бросаются в глаза, либо потому что они идейные, либо потому что занимают много места. В то же время, чтобы найти «мелкую» ошибку, нужно прочитать весь код, иногда несколько раз. Именно на поиски таких ошибок, обычно, тратится наибольшее количество преподавательского времени.
- «Мелкие» ошибки чаще всего встречаются в коде, который нужно посмотреть преподавателю на предмет ошибок, так как учащемуся проще их допустить, а также гораздо сложнее найти их самостоятельно.

- Такие ошибки также обладают свойством повторяться, ведь чем меньше ошибка, тем больше вероятность, что подобную повторит кто-либо другой.

### **Выводы по главе 1**

TBD