Оригинальная статья

https://doi.org/10.21869/2223-1560-2023-27-1-114-123

Автотестирование встраиваемой реконфигурируемой вычислительной системы

А. И. Мартышкин [1] ✉, И. А. Кирюткин [2], Е. А. Мереняшева [2]

[1] Пензенский государственный технологический университет
проезд Байдукова/ул.Гагарина, д. 1а/11, г. Пенза 440039, Российская Федерация

[2] Пензенский государственный университет
ул. Красная, д. 40, г. Пенза 440026, Российская Федерация

✉ e-mail: alexey314@yandex.ru

Резюме

Цель исследования. *Основная идея заключается в построении математической модели тестирования, которая объединяет различные аспекты встраиваемой реконфигурируемой вычислительной системы и ее взаимодействий. Эта модель обеспечивает эффективное представление тестовых сценариев и позволяет анализировать динамику реконфигурируемой вычислительной системы во время тестирования. В статье также обсуждаются методы формирования тестовых последовательностей, исходя из свойств конечного автомата.*

Методы. *Авторы предлагают представить автотест в виде конечного автомата Мили, где состояния служат для хранения информации о текущем состоянии встраиваемой реконфигурируемой вычислительной системы. Входным сигналом является взаимодействие с системой, а выходным сигналом – реакция системы на входной сигнал. Такой подход позволяет формализовать процесс тестирования и упростить анализ возможных проблем.*

Результаты. *В данной статье рассматривается применение теории автоматов в контексте автоматизированного тестирования встраиваемых реконфигурируемых вычислительных систем. Теория автоматов предоставляет эффективные методы и инструменты для анализа и моделирования дискретных динамических систем, что делает ее подходящей для решения задач автоматизированного тестирования.*

Заключение. *Результаты исследования показывают, что использование теории автоматов может существенно улучшить качество и эффективность автоматизированного тестирования встраиваемых реконфигурируемых вычислительных систем. Этот подход обеспечивает более глубокий анализ системы и позволяет обнаружить и предотвратить потенциальные проблемы, которые могут возникнуть в процессе ее работы.*

Ключевые слова: *автомат Мили; временные автоматы; конечные автоматы; теория автоматов; автоматизированное тестирование; автотест; встраиваемая реконфигурируемая вычислительная система.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Финансирование: Исследование выполнено за счет гранта Российского научного фонда № 21-71-00110, https://rscf.ru/project/21-71-00110/.

Для цитирования: Мартышкин А. И., Кирюткин И. А., Мереняшева Е. А. Автотестирование встраиваемой реконфигурируемой вычислительной системы // Известия Юго-Западного государственного университета. 2023; 27(1): 114-123. https://doi.org/10.21869/2223-1560-2023-27-1-114-123.

Поступила в редакцию 17.02.2023 *Подписана в печать 02.03.2023* *Опубликована 14.04.2023*

***

Введение

Автоматизированное тестирование встраиваемой реконфигурируемой вычислительной системы (РВС) может значительно упростить ее разработку и отладку, чем в случае, если бы тестирование производилось вручную. В качестве примера представим тестирование приложения мобильного телефона. Основные способы, средства и методы тестирования, представленные в данной статье, подойдут и для тестирования многоядерной РВС [1, 2, 3, 4]. С использованием автоматизированного тестирования программа выполняет проверку результатов своей работы при помощи специальных инструментов, что способствует существенному сокращению времени, затрачиваемого на проведение тестирования, и упрощает этот процесс [5, 6, 7]. В качестве примера протестируем мобильное приложение, которое складывает 2 числа.

Материалы и методы

Автотест можно описать как абстрактный автомат (АА), который представляет собой математическую модель дискретного устройства. АА характеризуется алфавитом состояний, в нашем примере это состояния РВС, входным алфавитом это действия над РВС, функциями переходов и выходов и начальным состоянием [8, 9, 10].

В нашем примере стартовое состояние будет состояние «S1».

Состояние «S1» представляет страницу «меню», «S2» – страница «калькулятор», «S3» – «инструкция».

Каждое состояние автомата имеет переходы и петли: переходы описывают переход с одной страницы на другую, а петли, тестовые действия на странице [9, 10] (рис.1).

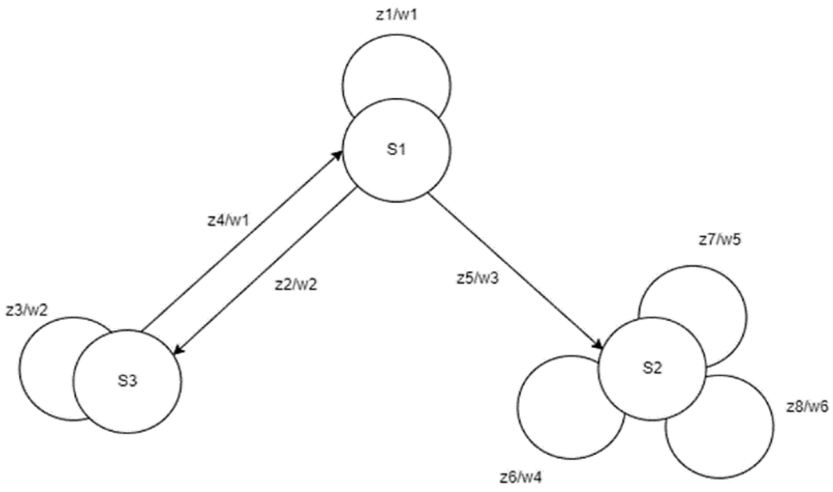

Рис. 1. Граф автотеста

Например, переход из состояния «S1» с сигналом «z2» ведёт в состояние «S3», что соответствует переходу к странице «инструкция», петля с входным сигналом «z1» соответствует проверки элементов на этой странице [1, 7, 11, 12].

В соответствии с графом была построена таблица переходов автомата (табл. 1).

Таблица 1. Таблица переходов

SM\zF	z1	z2	z3	z4	z5	z6	z7	z8
S1	w1	w2			w3			
S2						w4	w6	w5
S3			w2	w1				

СКУ

$S_1(t+1) = z_1 S_1 \vee z_4 S_3$;

$S_2(t+1) = z_5 S_2 \vee z_6 S_2 \vee z_7 S_2 \vee z_8 S_2$;

$S_3(t+1) = z_3 S_3 \vee z_2 S_1$.

СВФ:

$w_1(t) = z_1 S_1 \vee z_4 S_3$; $w_2(t) = z_3 S_3 \vee z_2 S_1$;

$w_3(t) = z_5 S_1$; $w_4(t) = z_6 S_2$;

$w_5(t) = z_8 S_2$;

$w_6(t) = z_7 S_2$; [3, 8, 9, 10, 13].

Сравнительный анализ сетей Петри и автоматов для автоматического тестирования вычислительных систем позволяет выявить ключевые различия между этими двумя подходами.

Сети Петри представляют собой графическую модель, состоящую из вершин, переходов и связей между ними. Они предоставляют наглядное представление параллелизма и синхронизации процессов. В то время как абстрактные автоматы представляют собой модель состояний и переходов между ними. В случае автоматов, структура модели представляет собой граф, вершины которого обозначают состояния, а ребра – переходы между состояниями. Сравнительная характеристика представлена в табл. 2.

Абстрактные автоматы представляют собой простой и понятный подход для моделирования и автоматического тестирования вычислительных систем. Они позволяют легко описывать последовательность действий пользователя и предсказуемо анализировать возможные состояния системы. Хотя Сети Петри обладают высокой выразительностью и могут быть полезны в некоторых случаях, они могут быть избыточными и сложными для анализа при тестировании вычислительных систем. Вместо этого, они лучше подходят для моделирования и тестирования сложных многопоточных и параллельных систем с множеством взаимодействующих компонентов.

Таблица 2. Сравнение автоматов и сетей Петри

Функционал	Сети Петри	Автоматы
Моделирование	Основным преимуществом сетей Петри является их способность представлять параллелизм, что может быть полезно для моделирования и анализа многопоточных приложений в автотестировании	Автоматы, будучи дискретными моделями, имеют ограниченные возможности для представления параллелизма, но хорошо подходят для реализаций последовательных действий
Визуализация	Сети Петри могут показать сложные взаимодействия и зависимости между состояниями системы, что может быть полезно при тестировании сложных многопоточных систем	Автоматы также предлагают визуальное представление, но оно может быть менее интуитивным и наглядным для представления параллелизма. Для описания последовательных шагов теста автоматы более подходят автотестированию
Анализ	Сети Петри позволяют проводить формальный анализ и верификацию, что может быть полезно для выявления ошибок и проблем в автотестировании. Однако анализ сетей Петри может быть сложным	Автоматы также позволяют проводить формальный анализ, но в отличие от сетей Петри, анализ автоматов может быть проще и менее затратным в некоторых случаях. Что удобнее использовать в автотестировании
Применимость к автотестированию	Сети Петри хорошо подходят для автотестирования систем с параллельными процессами, а также для анализа сложных зависимостей между состояниями. Однако из-за их сложности, они могут потребовать больше времени на разработку и анализ	Автоматы хорошо подходят для автотестирования простых и детерминированных систем. Их анализ может быть проще и быстрее, чем у сетей Петри. Однако для представления параллелизма они могут быть недостаточно подходящими

Для реализации автотестов используются следующие технологии:

Java – язык программирования.

Maven – фреймворк для автоматизации сборки проектов.

TestNG – фреймворк для создания отчетов тестирования.

Appium – среда автоматизации тестирования приложений на различных платформах, таких как iOS, Android и Windows.

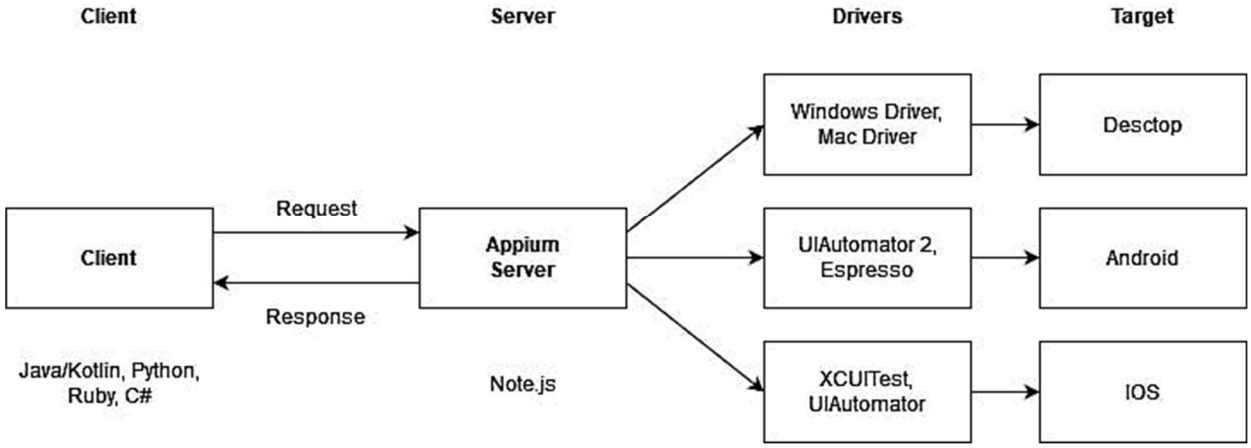

Рис. 2. Принцип работы Appium

Appium – это HTTP-сервер, который работает по следующему принципу (рис. 2):

- Сервер получает запрос на подключение от клиента.
- Сервер прослушивает команды.
- Клиент делает запрос к серверу.
- Сервер взаимодействует с различными драйверами платформы.
- Драйверы взаимодействуют с нативными приложениями, и выполняется необходимое действие в приложении.
- Сервер возвращает HTTP-ответ.
- Каждое действие в Appium происходит в контексте сеанса.
- Сеанс начинается с объекта Desired capabilities, чтобы указать точный тип соединения, которое мы хотим, и какое поведение нас больше всего интересует [14].

Результаты и их обсуждение

Приложение, написанное на языке Dart во фреймфорке flutter (рис. 4), представляет простой калькулятор для сложения двух чисел.

Для взаимодействия с этим приложением будем использовать flutter driver. Драйвер должен различать элементы приложения, поэтому в каждом элементе приложения пропишем «ключ» – некий идентификатор элемента [15, 16].

```
TextField(
  key: Key('NumberOne'),
  keyboardType: TextInputType.number,
      )
```

В приведенном примере текстовому полю присвоен ключ NumberOne, какое именно использовать название ключа не важно, главное, чтобы в приложении они не повторялись.

Рис. 3. Страница "Калькулятор" приложения

Сам автотест представляет собой проект Maven, на языке Java. В файле pom.xml проекта прописываем зависимости для appium и testNg, testNg необходим для формирования отчета по пройдённым тестам [2].

В проекте взаимодействие с приложением происходит через драйвер, AndroidDriver<MobileElement> driver. Для инициализации драйвера необходимо задать ему свойства, среди них имя устройства, версия операционной системы (ОС) устройства, имя пакета тестируемого приложения и другие. Так же указываем путь к локальному хосту, с которого запущен аппиум сервер.

```
flutterCapabilities.setCapability ("appium:platformVersion", "11");
flutterCapabilities.setCapability ("appium:reportDirectory", "html");
driver = new AndroidDriver<MobileElement> (new URL ("http://127.0.0.1:4723/wd/hub"), flutterCapabilities);
```

После того, как мы инициализировали драйвер, можем перейти непосредственно к автотесту [2, 14, 17, 18]. Чтобы обращаться к элементам, используем скрипт flutterWait и указываем параметры, по которым мы можем работать с элементами, например, для того чтобы найти элемент в приложении, используем

```
driver.executeScript("flutter:waitFor", find.byValueKey(valueKey), timeout),
```

где метод *find.byValueKey(valueKey)* указывает, что элемент ищем по ключу *valueKey*, например, можем передать ключ *NumberOne*. Упомянутый выше функционал тестов не ограничивается только поиском элементов по ключам. Так же можем использовать поиск по тексту, пролистывание, нажатие на элемент, ввод текста в текстовые поля. Функционал покрывает все действия, которые человек может повторить вручную [6].

Подход в организации проекта будет таким (рис. 6). Методы для нахождения элементов будут находиться в суперклассе, от которого будут наследоваться классы, которые будут соответствовать страницам нашего приложения для теста PBC. Далее в классе теста будут использоваться методы объектов наших страниц. Каждый метод будет представлять собой действие, по функционалу аналогичное действию человека [19, 20, 21, 22].

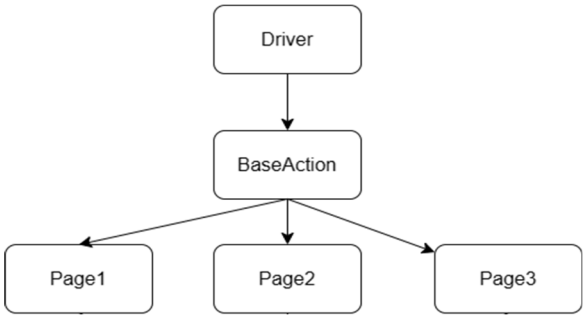

Рис. 4. Иерархия классов теста

Итоговый тест будет выглядеть так:

```
public class TestFlutterApplication
extends BaseTest {
    public void testApp()
    {
      MenuPage menuPage= new MenuPage();
      ManualPage manualPage=new Man-
ualPage();
      CalculatorPage calculatorPage=
new CalculatorPage();
      menuPage.checkElements()
            .goToManual();
      manualPage.checkElements()
            .goToMenu();
      menuPage.goToCalculator()
      calculatorPage.setFirstNumber()
                    .setSecondNumber()
                    .getSum();
    }
}
```

Таким образом, тест [5] представляет собой последовательность действий, понятных каждому человеку, а полный код тестов спрятан в суперклассах страниц.

Количественная оценка предложенного метода:

1. Экспериментальным путем получено, что время работы автотеста значительно быстрее ручного тестирования, все время уходит только на составление автотеста. Для приведенного в исследовании теста время выполнения ручного тестирования инженером достигает 30 секунд, автотеста – 2 секунды. На рис. 5 приведено сравнение производительности выполнения тестирования при разных подходах.

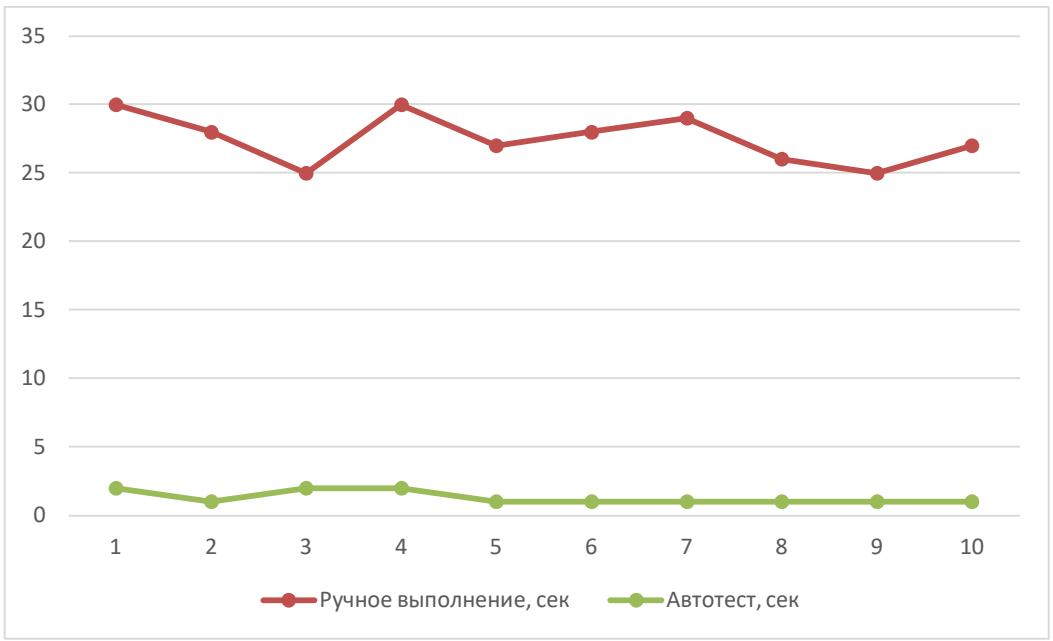

Рис. 5. Сравнение производительности подходов тестирования

Понятность и наглядность автотестов позволяет легко воспроизводить тесты, созданные с использованием предложенного подхода по сравнению с другими методами. Более понятные и наглядные тесты упрощают их использование.

Гибкость тестирования упрощает внесение изменений в программное обеспечение и тесты, что снижает время на разработку автотестов под новые изменения.

Эти метрики показывают эффективность и преимущества данного метода автоматического тестирования на основе абстрактных автоматов.

Выводы

В данной статье мы представили автотест мобильного приложения в виде конечного автомата, который в свою очередь визуально представляет работу автотеста, что служит примером теста РВС.

Список литературы

1. Волчихин В.И., Вашкевич Н.П., Бикташев Р.А. Модели событийных недетерминированных автоматов представления алгоритмов управления взаимодействующими процессами в многопроцессорных вычислительных системах на основе использования механизма монитора // Известия высших учебных заведений. Поволжский регион. Технические науки. 2013. № 2 (26). С. 5-14.

2. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Использование конечных автоматов для тестирования программ // Программирование. 2000. Т. 26. №. 2 С. 61–73.

3. Formalized Description of Cyber-Physical Systems Models Using Temporary Non-Deterministic Automata / D.A. Trokoz, K.A. Zabrodina, V.S. Safronova, M.D. Veselova, M.P. Sinev, M.A. Mitrohin, E.I. Gurin, T.Y. Pashchenko // Journal of Physics: Conference Series. II International Scientific Conference on Metrological Support of Innovative Technologies (ICMSIT II-2021). Krasnoyarsk, 2021. Vol. 1889. №. 2. 22068 p. IOP Publishing.

4. Гузик В.Ф., Каляев И.А., Левин И.И. Реконфигурируемые вычислительные системы. Таганрог.: Южный федеральный университет, 2016. 472 с.

5. Котенко А.П., Букаренко М.Б. Моделирование конечными автоматами систем массового обслуживания с различимыми каналами // Известия Самарского научного центра Российской академии наук. 2014. Т. 16. № 4-2. С. 318-321.

6. Дубинин В.Н., Дроздов Д.Н. Проектирование и реализация систем управления дискретными событийными системами на основе иерархических модульных недетерминированных автоматов (Ч. 1. Формальная модель) // Известия высших учебных заведений. Поволжский регион. Технические науки. 2016. № 1 (37). С. 28-39.

7. Симанков В.С., Толкачев Д.М. Моделирование сложных объектов в режиме реального времени на основе сетей Петри // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2012. № 4 (110). С. 202-209.

8. Методика преобразования темпорального конечного автомата в СП-модель / Д.А. Трокоз, Р.А. Бикташев, М.П. Синев, М.С. Федяшов, Н.Н. Шеянов // XXI век: итоги прошлого и проблемы настоящего плюс. 2020. Т. 9. № 3 (51). С. 45-49.

9. Соловьев В. В. Минимизация конечных автоматов Мили путем использования значений выходных переменных для кодирования внутренних состояний // Известия Российской академии наук. Теория и системы управления. 2017. № 1. С. 98-106.

10. Твардовский А. С., Евтушенко Н. В., Громов М. Л. Минимизация автоматов с таймаутами и временными ограничениями // Труды Института системного программирования РАН. 2017. Т. 29. №. 4. С. 139-154.

11. Берёза А. Н., Ляшов М. В. Эволюционный синтез конечных автоматов // Известия ЮФУ. Технические науки. 2011. №7. С 210–217.

12. Темпоральный анализ киберфизических систем с использованием теории автоматов / Д.А. Трокоз, Н.В. Исхаков, М.П. Синев, М.А. Митрохин, Н.О. Сивишкина // XXI век: итоги прошлого и проблемы настоящего плюс. 2019. Т. 8. № 3 (47). С. 113-117.

13. Захаров Н.Г., Рогов В.Н. Синтез цифровых автоматов. Ульяновск, 2003. 135 с.

14. Сетевая модель языка "Т" на основе цветных безопасных иерархических рекурсивных сетей Петри / Д.В. Пащенко, Д.А. Трокоз, Г.В. Мартяшин, К.С. Максимова, Е.А. Бальзанникова // Известия высших учебных заведений. Поволжский регион. Технические науки. 2015. № 3 (35). С. 25-35.

15. Tashildar A. et al. Application development using flutter // International Research Journal of Modernization in Engineering Technology and Science. 2020. Vol. 2. № 8. P. 1262-1266.

16. Денисов А. А. Современные средства разработки мобильных приложений // Вестник Воронежского института высоких технологий. 2019. № 2. С. 64-67.

17. Dai W., Dubinin V., Vyatkin V. Automatically Generated Layered Ontological Models for Semantic Analysis of Component Based Control Systems //IEEE Transactions on Systems, Man, and Cybernetics: Systems (IEEE Publishing). 2013. Vol. 9. № 4. P. 2124–2136.

18. Generalized structural models of complex distributed objects / M. Yu. Mikheev, T.V. Zhashkova, A. B. Shcherban, A. K. Grishko, I. M. Rybakov // Proceedings of 2016 IEEE East-West Design and Test Symposium, EWDTS 2016. Yerevan: Institute of Electrical and Electronics Engineers Inc., 2017. p. 7807742. DOI 10.1109/EWDTS.2016.7807742. EDN YVGCCT.

19. Рыбалко М. А., Иванова Е. А. Тестирование программного обеспечения, методы тестирования // Информационное общество: современное состояние и перспективы развития. Краснодар, 2017. С. 320-322.

20. Караваева О. В., Борисова К. В. Разработка системы автоматизированного тестирования // Вестник науки и образования. 2018. № 13 (49). С. 45-48.

21. Надыкто М. О., Белим С. В. Разработка автотестов для обеспечения качества программного обеспечения // Образование. Транспорт. Инновации. Строительство. Омск, 2022. С. 588-594.

22. Лагарникова А. В. Алгоритм построения автотестов на основе часто используемых тестовых сценариев в мобильном приложении // Вестник науки. 2019. Т. 1. № 2 (11). С. 155-158.

Информация об авторах

Мартышкин Алексей Иванович, кандидат технических наук, доцент, заведующий кафедрой «Программирование», Пензенский государственный технологический университет, г. Пенза, Российская Федерация, e-mail: alexey314@yandex.ru, ORCID: http://orcid.org/0000-0002-3358-4394

Кирюткин Илья Алексеевич, студент кафедры «Вычислительная техника», Пензенский государственный университет, г. Пенза, Российская Федерация, e-mail: kiryutkin.02@mail.ru, ORCID: http://orcid.org/0009-0001-4355-0796

Мереняшева Елизавета Андреевна, студент кафедры «Вычислительная техника», Пензенский государственный университет, г. Пенза, Российская Федерация, e-mail: Lizamerenyasheva@gmail.com, ORCID: http://orcid.org/0000-0003-3744-4521