МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра Кафедра информатики и программирования

РАЗРАБОТКА КОМПЛЕКСА АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ РАЗРАБОТАННГО ТРАНСЛЯТОРА ИЗ ЯЗЫКА ПРОГРАММИРОВАНИЯ КУМИР В ЯЗЫК C++

КУРСОВАЯ РАБОТА

студента 1 курса 173 группы

направления 02.03.04 — Математическое обеспечение и администрирование информационных систем

КНиИТ

Пронина Антона Алексеевича

Научный руководитель

старший преподаватель _____ Е. Е. Лапшева

Заведующий кафедрой

к. ф.-м. н. _____ М. В. Огнева

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Методы, средства и технологии 5
1.1 Методы и средства автоматизированного тестирования 5
1.2 Транслятор с языка КуМир в язык C++ 9
2 Реализация системы тестирования 15
2.1 Процесс тестирования 15
2.2 Тесты 15
2.3 Программа-тестер 17
ЗАКЛЮЧЕНИЕ 20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22

ВВЕДЕНИЕ

В современном школьном образовании среди прочих можно выделить следующие тенденции: растущий интерес к изучению робототехники; глубокая интеграция языка программирования КуМир в школьную информатику и государственную итоговую аттестацию по этому предмету.

В связи с этим будет востребована система программирования роботов с использованием этого языка программирования. Ключевой составной частью этой системы является транслятор программы с языка КуМир [1] в язык C++. Трубуется протестировать разработанный транслятор, провести поиск ошибок и устранить их.

Создание средств и систем обеспечения надежности ПО такого класса как компиляторы, представляет собой весьма сложную задачу, особенно, если речь идет о тестировании компиляторов. Тестирование компиляторов - это необходимый этап в разработке языков программирования и компиляторов, поскольку компиляторы являются ключевыми инструментами для трансформации исходного кода в машинный код, который может выполняться на целевой платформе. Наличие надежного и качественного компилятора обеспечивает уверенность в правильности работы программ и уменьшает риски возникновения ошибок во время выполнения программы. Тестирование компиляторов необходимо для гарантии совместимости программного кода на различных платформах и операционных системах. Кроме того, тестирование компиляторов помогает в оценке работоспособности и эффективности программного обеспечения на целевой платформе. В целом, тестирование компиляторов является важным этапом в процессе разработке программного обеспечения, который может помочь в достижении более высокого качества и уверенности в работе программного кода.

Актуальность транслятора заключается в снижении входного порога в

область робототехники как со стороны ученика, предоставляя возможность, используя полученные навыки программирования в системе КуМир, заниматься разработкой роботов, так и со стороны преподавателя, уменьшая затраты на приобретение программных и аппаратных средств разработки. В качестве аппаратной платформы данной задачи был выбран электронный конструктор и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов [2] - платы Arduino. Ввиду низкой стоимости устройств, периферийных модулей, простоты разработки аппаратных устройств на базе этих плат, высокой модульности систем и их высокой распространенности среди робототехников.

Цель работы - разработка комплекса автоматизированного тестирования из языка программирования КуМир в язык C++.

Для выполнения поставленной цели, требуется выполнить следующие задачи:

- анализ подходов, инструментов и ПО, использующихся при автоматизации тестирования;
- разработка набора тестов и документации для транслятора;
- разработка системы автоматизированного тестирования.

1 Методы, средства и технологии

1.1 Методы и средства автоматизированного тестирования

Автоматизированное тестирование - это процесс тестирования, в котором используются специальные программные средства и скрипты, которые могут автоматически создавать и проводить тесты на соответствие функциональным требованиям. Автоматизированный тест позволяет быстро и эффективно производить тестирование без физического участия тестировщика. Это позволяет повысить качество продукта, уменьшить количество ошибок и рисков при релизе, а также быстро и точно проводить регрессионное тестирование после внесения изменений в проект. Автоматизированное тестирование может уменьшить время, необходимое для тестирования, а также заставить взглянуть на проблемы, с которыми можно столкнуться во время процесса разработки. В качестве основных видов автоматизированного тестирования можно выделить следующие:

- Unit-тестирование: тестирование отдельных компонентов программного обеспечения, таких как функции, классы и модули;
- интеграционное тестирование: тестирование взаимодействия между отдельными компонентами;
- функциональное тестирование: тестирование функциональности программного обеспечения;
- конфигурационное тестирование: тестирование работоспособности системы при изменении настроек и конфигураций;
- нагрузочное тестирование: тестирование производительности системы при различных нагрузках;
- тестирование безопасности: тестирование на возможность атак и нарушений безопасности системы;

• тестирование совместимости: тестирование возможности программного обеспечения работать с различными операционными системами, браузерами и устройствами;

• тестирование пользовательского интерфейса: тестирование удобства и функциональности интерфейса программного обеспечения;

• автоматизированное тестирование: использование специальных средств для создания скриптов тестирования и автоматизации процесса тестирования.

В зависимости от платформы, применяются различные инструменты и методы тестирования. в качестве тестируемого объекта выступает транслятор из языка программирования КуМир в язык C++.

При создании системы тестирования, требуется максимально уменьшить количество внешних эффектов, создаваемых окружением. Для унификации среды для запуска системы тестирования используется технология контейнеризации и Docker.

Оконтейнеризация (или контейнеризация) - это технология виртуализации, которая позволяет запускать и управлять различными приложениями в изолированной среде - контейнере. Контейнеры являются автономными, портативными и легковесными единицами, которые могут быть запущены в любой совместимой с контейнеризацией среде.

Контейнеры позволяют упаковывать приложения и их зависимости в единую единицу, которая может быть развернута и запущена в любом месте, без необходимости инсталляции конкретных зависимостей на хост-системе. Контейнеры дают возможность разработчикам и DevOps-инженерам значительно ускорить процесс разработки, тестирования и распространения приложений.

Контейнеры используют ядро операционной системы хоста и создают определенное окружение, в котором приложение запускается и работает. Это

облегчает многие процессы, такие как масштабирование, управление зависимостями и обновлениями, управление средой и конфигурацией приложения.

Docker - это свободное программное обеспечение, которое позволяет упаковывать, доставлять и запускать приложения в контейнерах. Это популярная технология контейнеризации, которая позволяет разработчикам быстро и легко создавать, тестировать и запускать приложения в различных средах. Docker также предоставляет инструменты для управления контейнерами, включая CLI и API. Это позволяет управлять контейнерами и приложениями из командной строки или с помощью API, что делает автоматизацию и масштабирование процесса разработки и развертывания более эффективным. Docker - это инновационная технология, которая упрощает процессы разработки, тестирования и запуска приложений и может использоваться в любом окружении. Она становится все более популярной в мире разработки приложений благодаря своей надежности, безопасности и высокой скорости запуска.

С точки зрения тестирования, транслятор - просто программа, принимающая информацию на вход и в результате работы имеющая определенный выход. Можно выделить 4 основных способа тестирования компиляторов:

1. наборы небольших статических тестов (regression, unit и т.п.). Это наборы тестов программного обеспечения, которые написаны до запуска приложения и проверяют его на соответствие требованиям и описанию функций. Эти тесты используются для проверки корректности работы приложения до его использования реальными пользователями.;
2. приложения-тестеры. Это приложение позволяет создавать тестовые сценарии, выполнять автоматические тесты и генерировать отчеты о результатах тестирования. Обычно приложения-тестеры используются во время автоматического тестирования, когда тесты выполняются без участия человека. Это может быть полезно для повторяемых тестов, которые нужно

проводить на протяжении всего периода разработки приложения;

3. coverage-тестирование. Это метод тестирования программного обеспечения, который позволяет оценить, насколько хорошо тесты покрывают код программы. Он используется для определения того, сколько кода было протестировано путем анализа выполнения тестов и сравнения с фактическим кодом программы. Результаты coverage-тестирования определяют, какие участки кода были покрыты тестами и какие нет. Если некоторые участки кода не были покрыты тестами, то разработчики смогут сосредоточиться на их тестировании и повышении надежности программного обеспечения.;
4. fuzzing-тестирование. Это метод тестирования программного обеспечения, который заключается в автоматической генерации большого количества случайных или полу-случайных входных данных, и последующем анализе ответов на эти входные данные. Целью такого тестирования является обнаружение ошибок и уязвимостей в программном обеспечении, которые могут привести к потенциальным угрозам безопасности или нарушению работоспособности приложения [3].

Среди готовых решений для тестирования ПО можно выделить следующие:

- IBM Rational Functional Tester[4] предоставляет функции автоматического тестирования для функционального, регрессионного тестирования, тестирования графических пользовательских интерфейсов и тестирования, ориентированного на данные. Данная программа не распространяется бесплатно, для использования необходимо приобрести лицензию;
- IBM Rational Test Workbench[5] так же распространяется по платной лицензией, с пробным периодом бесплатного использования в тридцать дней;
- система автоматизированного тестирования TestComplete[6], предостав-

ляющая в том числе инструмент для регрессионного тестирования.

На этапе анализа существующего ПО для тестирования, было выяснено, что ни 1 из рассмотренных вариантов не сможет быть применен в виду чрезмерно объемного интерфейса, а также отсутствия бесплатной подписки.

Транслятор из языка программирования КуМир в язык программирования C++ - разработка с открытыми исходными кодами. На данный момент, язык программирования КуМир поддерживает функциональный стиль программирования без создания пользовательских структур или классов. Ввиду наличия исходных кодов транслятора и ограниченности набора инструкций языка КуМир, было решено разработать автоматизированную тестовую среду с набором статических тестов.

1.2 Транслятор с языка КуМир в язык C++

Приложение разрабатывается на языке C++ с использованием фрейморвка QT 4 или 5 версии, а также системы автоматизации сборки программного кода CMake. Для сборки проекта на операционной системе Windows, требуется программа-сборщик MS Build 2012, устаревший на данный момент и труднодоступный для скачивания, поэтому было предпринято решение разрабатывать приложение на операционной системе Linux Mint. Для сборки проекта требуется установить ряд дополнительных библиотек, среди которых: ZLib, Boost и интерпретатор языка программирования Python.

Проект состоит из ряда библиотек, используемых внутри проекта, в том числе - AST-дерево, библиотека для работы с низкоуровневой виртуальной машиной, библиотеки для рендеринга pdf-документов, . . . ; плагинов, а также исполнителей и сред разработки, состоящих из файлов с описанием использующихся методов, процедур и сущностей и файлов с описанием слоя представления данных частей. Ввиду большого объема приложения, высокого уровня

связности элементов друг с другом и устаревших библиотек, используемых для разработки, сборка проекта осуществляется при помощи командной строки.

В плагине kumirCodeGenerator осуществляется трансляция и компиляция программного кода с языка КуМир в язык низкоуровневой виртуальной машины. По имеющемуся файлу с исходным кодом происходят лексический и синтаксический анализы, в случае, если они завершились успешно и не было выявлено ошибок, трансляция продолжается. В результате лексического анализа имеется набор лексем кода исходного языка программирования, использующийся для синтаксического анализа. В результате синтаксического анализа программа создает набор токенов для генерации по ним дерева разбора. Вычисленное дерево разбора усекается до AST-дерева, для трансляции кода. На этом этапе происходит наращивание кода на выходном языке по имеющемуся дереву разбора. Описанные выше этапы осуществляются при помощи вызова функций сторонних модулей.

Для разработки транслятора на основе измененного и переработанного программного кода плагина kumirCodeGenerator был создан модуль arduinoCodeGenerator. Основные изменения затрагивают сущность Generator, содержащую основной объем операций по обработке данных AST-дерева и наращиванию программного кода по нему на выходном языке.

Поскольку в изначальном варианте программный код транслировался в язык виртуальной низкоуровневой машины, был кардинально переработан список команд. Список команд для трансляции с языка программирования КуМир в язык программирования C++ приведен в таблице 2.

Были изменены инструкции для трансляции циклов, условий, вызова и объявления функций. Были добавлены инструкции для объявления переменных и констант. Блок операций подвергся минимальным изменениям — были добавлены инструкции инкремента и декремента, а также оператор присваива-

Таблица 1 – Список команд, используемый для трансляции с языка программирования КуМир в язык программирования C++

Название операции	Код операции
ForLoop	0
WhileLoop	1
VAR	2
ARR	3
FUNC	4
CONST	5
IF	6
ELSE	7
SWITCH	8
CASE	9
INPUT	10
OUTPUT	11
BREAK	12
END_ARG_DELIM	13
END_FUNC_DELIM	14
END_ARR_DELIM	15
STR_DELIM	16
END_ST_DELIM	17
END_VAR_DELIM	18
END_ST_HEAD_DELIM	19
RET	20
SUM	21
SUB	22
MUL	23
DIV	24
POW	25
NEG	26
AND	27
OR	28
EQ	29
NEQ	30
LS	31
GT	32
LEQ	33
GEQ	34
ASG	35
DCR	36
INC	37

ния.

Также изменилась основная сущность, используемая при трансляции — сущность инструкции выходного языка. Были удалены поля для хранения данных о регистре, спецификации строки, спецификации модуля и были добавлены поля для хранения имени операнда и типы операнда, если присутствует.

Трансляция циклов Существует 4 типа циклов в языке программирования КуМир: стандартный для языков программирования высокого уровня цикл, выполняющийся некоторое количество раз, определяемое набором элементов в указанном диапазоне “нц для“, цикл с предусловием “нц пока“, цикл, выполняющийся n раз “нц для n раз“, а также цикл, выполняющийся постоянно до остановки исполнения “нц всегда“. При трансляции в низкоуровневый язык, этап трансляции циклов требовал одной инструкции — “LOOP“, а также тела операций, выполняющихся до данной метки. В начале транслировалось тело цикла, в случае цикла с предусловием до тела транслировалось условие исполнения, на последнем этапе трансляции добавлялась инструкция “LOOP“, объявляющую метку завершения определения цикла.

Для трансляции циклов в язык C++ были добавлены инструкции для соответствующих типов циклов — ForLoop, WhileLoop. Циклы “ нц пока“и “ нц всегда“ определяются при помощи инструкции WhileLoop, “нц для“ и “нц для n раз“ - при помощи ForLoop.

В начале трансляции цикла любого типа указывается заголовок, определяющий тип цикла и возможные предусловия. В случае цикла “нц пока“, заголовок содержит условие окончания работы, состоящее из ряда выражений.

После трансляции заголовка инструкции цикла транслируется тело цикла. Трансляция завершается инструкцией END_ST_DELIM.

Трансляция подвыражений Трансляция подвыражений осуществляется при помощи метода calculate. Любое подвыражение представляется бинарным деревом, в вершине которого находится операнд, а на листьях - константы или переменные. Для корректной трансляции подвыражений требуется разобрать дерево подвыражения снизу вверх — найти узел, листья которого не содержат дальнейшей вложенности, затем добавить в стек инструкций левый лист узла, затем сам узел, в конце — правый лист. Метод calculate используется во множестве мест и содержит логику для трансляции переменных, констант, вызовов функций и подвыражений. Поскольку метод является рекурсивным, невозможно добавить дополнительный блок операций трансляций, описанный выше, в тело метода calculate, поэтому было принято решение вынести логику данного метода в новый - innerCalculation, превратив метод исходный в обертку, куда и была добавлена вспомогательная логика.

Трансляция условных выражений Изначально, трансляция условных конструкций состояла из определения количества подвыражений транслируемого выражения и поиска ошибок. На каждую конструкцию “если то“ и “иначе“ добавлялась инструкция безусловного перехода, изменяя ход исполнения программы. Для установки места программы низкоуровнего языка, куда совершался переход используется регистр IP.

В разработанной реализации трансляция условных выражений повторяет трансляцию цикла с предусловием за исключением первой инструкции — вначале, в стек инструкций добавляется инструкция “IF“ или “ELSE“, указывающая транслятору на объявление соответствующего блока. Процесс повторяется пока сущность, содержащая данные об условном выражении не опустеет.

Трансляция блоков ‘выбор‘ происходит следующим образом: вначале в стек заносится инструкция с кодом “SWITCH“, далее добавляется переменная,

значения которой перебираются и на каждый блок ‘при условие‘ добавляется инструкция “CASE“ и константа, хранящая значение переменной. В случае наличия метки “иначе“ в блоке “выбор“, добавляется инструкция “CASE“ без константы, транслирующаяся в инструкцию “default“ в коде выходного языка.

Трансляция переменных и констант В исходной версии при трансляции констант в выражениях или при инициализации, в стек виртуальной машины заносился индекс переменной или константы среди всех встреченных при трансляции, извлекаемый по ссылке на этапе исполнения. В разработанной реализации в случае использования переменной в стек добавляется инструкция “VAR“, хранящая ссылку на название переменной и на тип, в случае объявления. Трансляция констант начинается с занесения в стек инструкции “CONST“, хранящей индекс значения и тип константы в случае инициализации.

2 Реализация системы тестирования

2.1 Процесс тестирования

Под тестированием компилятора понимается сборка транслятора из исходных кодов, трансляция тестовой программы и сравнение результатов работы транслятора с ожидаемым результатом. Было решено автоматизировать процесс тестирования, для повышения качества и снижения сложности процесса.

Т.к. язык программирования и среда КуМир - кроссплатформенные проекты, в начале процесса автоматизации стоило решить вопрос с внешними условиями - операционной системой, набором используемых пакетов и библиотек, . . . , т.о. первым шагом к автоматизации тестирования явилось создание централизованной среды для сборки компилятора, а именно разработка dockerfile файла, содержащий набор инструкций для сборки транслятора из исходных кодов. В качестве операционной системы была выбрана Ubuntu, ввиду широкой распространенности, большого сообществу разработчиков и активной поддержке пакетов с данной системой в репозитории docker.

Далее встал вопрос автоматизации запуска транслятора. Для решения этой проблемы был разработан скрипт на языке Python 3.

Для автоматизации запуска тестовой системы в репозитории разрабатываемого транслятора были настроены службы автоматического запуска(Github Actions) при отправке изменений и создании запроса на внесение изменений в целевую ветвь.

2.2 Тесты

Для тестирования были разработаны набор тестов, покрывающие основные инструкции языка программирования КуМир, а именно: работа с переменными(инициализация, присваивание), арифметические и логические выражения, ветвления, циклы, а также функции и процедуры. Тестирование построено

на принципе сравнения ожидаемого результат с результатом работы транслятора. Тесты сгруппированы по подмножествам инструкций исходного языка.

Рисунок 1 — Группы тестов

Группа состоит из секций, представляющих отдельные инструкции. Для создания тестовой секции, например, для тестирования трансляции функций, необходимо 2 файла - файл с расширением .kum на исходном языке программирования, а также файл с расширением .exp - файл с предполагаемым результатом работы транслятора, содержащий программу на языке C++. Пример секции тестов приведен на рисунке 2.

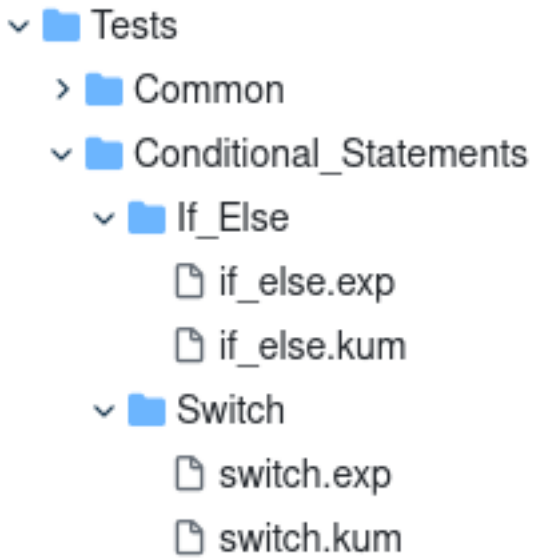

Рисунок 2 — Пример секции тестов

Под тестом транслятора понимается 2 файла, содержащие 1 или несколько

инструкций на исходном и выходном языках программирования. На данный момент разработано чуть больше 200 тестов.

2.3 Программа-тестер

Для автоматизации тестирования было разработано консольное приложение-тестер транслятора. В качестве языка программирования был выбран Python 3. Для работы с программой был разработан ряд аргументов командной строки, а также справка. Список аргументов командной строки приведен ниже:

```
[-h] [--help] - show help.
[-tr] [--translator] ["the path to pre-builded kumir2 to arduino translator
instance"] - show app what translator instance to use.
[-t] [--path-to-tests] ["the path to tests folder"] = show app
  the folder with test files.
[-o] [--output] ["the path to log file"] - show app where to store test logs.
[-d] [--duplicate] - duplicate output to console.
[-ss] [--skip-successfull] - skip open log info about succesfully completed tests.
[-sf] [--skip-failed] - skip open log info about failed tests.
[-swe] [--skip-without-expectation] - skip open log info about
 tests for which the file with expectations was not found.
[-sce] [--skip-with-compiler-error] - skip open log info about
 tests ended with compiler error.
[-b] [--brief] - skip open log info about all tests.
```

Было решено разработать программу гибкой и информативной. Для работы необходимы 2 аргумента - –translator(путь к транслятору), а также –path-to-tests(путь к папке с тестами). По умолчанию результат тестирования записывается в файл, также есть возможность продублировать вывод в консоль. Существует 4 состояния, отражающие результат работы компилятора:

1. успех;
2. провал;

3. ошибка компиляции;
4. не хватает файла с ожиданиями.

Состояния 3 и 4 могут быть получены по причине некорректного содержания тестовых файлов - либо написанная на языке программирования КуМир программа имеет ошибки и не может быть скомпилирована, либо не был найден файл с расширением .exp, содержащий ожидания работы компилятора. В других случаях, текст сообщения с результатами тестирования содержит предложение вызвать команду vimdiff с путями до файлов и прореферировать результат. Можно посмотреть как три файла сразу (исходный код, ожидаемый результат, результат работы компилятора), так и только ожидания и результат работы. Пример сообщения о результатах работы приведен на рисунке 3.

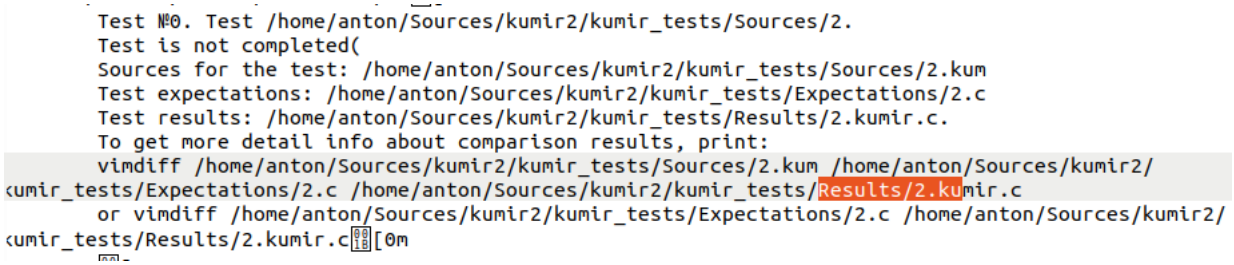

Рисунок 3 — Пример сообщения с возможностью подробно посмотреть результаты тестирования

В начале вывода информации приведена краткая статистика - сколько тестов завершилось и с каким состоянием. Объем вывода можно фильтровать и убирать подробную информацию о сообщениях в состояниях 1-4, либо полностью отключить подробный вывод информации, добавив флаг -b. При дублировании вывода в консоль, в зависимости от состояния, сообщение будет выделено цветом. Пример краткого вывода по результатам тестирования приведен на рисунке 4.

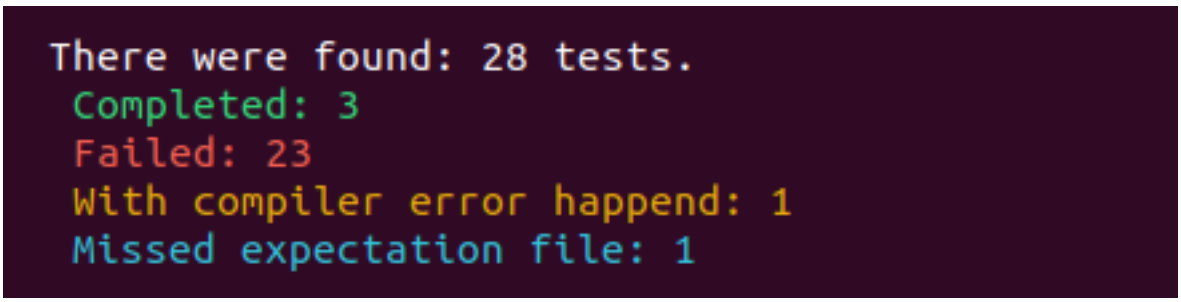

Рисунок 4 — Пример краткого вывода по окончанию тестирования

ЗАКЛЮЧЕНИЕ

В ходе тестирования разработанного транслятора с языка программирования КуМир в язык программирования C++ был выявлен ряд ошибок. Автоматизация процесса тестирования позволяет быстро расширять список тестов и увеличивать процент покрытия, влияющий на надежность и безотказность работы транслятора. Создание тестовых случаев позитивно сказывается на процессе разработки, ведь тестовые сценарии являются документацией. Автоматизация запуска программы-тестера при помощи Github Actions позволит быстрее отсеивать некачественный код, влияющий на работоспособность транслятора. В дальнейшем разработанная система тестов может быть доработана при помощи методов Fuzzing-тестирования и автоматизированной генерации тестовых файлов. Исходный код разработанной программы тестера и транслятора можно найти в открытом репозитории [7].

В рамках ВКР был был разработан транслятор с языка программирования КуМир в язык C++. Программа имеет ряд недостатков, которые предстоить исправить и список улучшений, которые планируется реализовать. Среди задач по улучшению транслятора можно выделить следующие:

- создание отдельного клиента среды программирования КуМир, специально для разработки роботов;
- добавление возможности прошивки робота из клиента;
- добавление инструмента выбора порта с подключенным роботом для прошивки;
- добавление настраиваемого алгоритма прошивки, определяющего роль результата трансляции в архитектуре программы для прошивки робота.

После реализации клиента среды программирования КуМир для разработки роботов и исправления ошибок транслятора найденных в ходе тестиро-

вания планируется спроектировать и разработать робота на базе аппаратного комплекса Arduino для опробации разработки в школах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Статья о КуМире на электронном образовательном портале Фоксфорд [Электронный ресурс] URL: $https://foxford.ru/wiki/informatika/$ sredaprogrammirovaniya-kumir

2 Статья о предназначении комплектов Arduino [Электронный ресурс] URL: $http://arduino.ru/$

3 Максименков Д.А., Рогов Р.Ю. Применение метода инструментирования тестовых программ при отладке оптимизирующих компиляторов. Вопросы радиоэлектроники, 2010, вып. 3, стр. 50-61

4 IBM. IBM Rational Functional Tester // IBM.

5 IBM. IBM Rational Test Workbench // IBM.

6 SmartBear. TestComplete Platform // SmartBear.

7 Репозиторий с исходным кодом транслятора [Электронный ресурс] URL:$https://github.com/CaMoCBaJL/kumir2/tree/translator_tests$

8 Система программирования Кумир 2.x А.Г.Кушниренко , М. А. Ройтберг , Д.В.Хачко, В. В. Яковлев [Электронный ресурс] URL:$http://roytberg.lpm.org.ru/pdfs/kumir2x_2015.pdf$

9 Леонов А.Г., Кушниренко А.Г. Методика преподавания основ алгоритмизации на базе системы «КуМир». М.: «Первое сентября», 2009.

10 Кушниренко А.Г., Рогожкина И.Б., Леонов А.Г. // Большой московский семинар по методике раннего обуч. информатике (ИТО-РОИ-2012): сб. докл. ПиктоМир: пропедевтика алгоритмического языка (опыт обучения программированию старших дошкольников). М.: Конгресс конференций ИТО-РОИ, 2012.