МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

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

УТВЕРЖДАЮ

Зав.кафедрой,

к. ф.-м. н.

_____ М. В. Огнева

ОТЧЕТ О ПРАКТИКЕ

студента 2 курса 273 группы КНиИТ

Пронина Антона Алексеевича

вид практики: производственная распределенная (научно-исследовательская работа)

кафедра: Кафедра информатики и программирования

курс: 2

семестр: 3

продолжительность: 18 нед., с 01.09.2023 г. по 15.01.2024 г.

Руководитель практики от университета,

старший преподаватель _____ Е. Е. Лапшева

Тема практики:«Разработка системы программирования “КуМир роботы“»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одним из наиболее интересных вопросов, требующих особого внимания в обучении информатике, является вопрос системы обучения программированию. Это связано с тем, что профессия специалистов в области информатики и информационных технологий в какой-то мере начинается со школы. Одним из прямых приложений программирования является робототехника.

Ввиду растущего интереса в сфере образования к обучению робототехнике в школе и широкой распространенности и глубокой степени интеграции современной образовательной системы с этим языком, появилась идея о разработке транслятора с языка КуМир в язык C++. Актуальность идеи 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 - 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.

Транслятор с языка КуМир в язык C++ стал отправной точкой процесса создания комплекса аппаратно-программных и методических средств изучения робототехники в школе с использованием языка программирования КуМир. Важным аспектом комплекса является доступность и удобство изучения робототехники. Для повышения доступности изучения, было принято решение разработать отдельный клиент системы программирования КуМир “КуМир Ро-

боты”, направленный на изучение робототехники на основе плат Arduino.

Цель работы - разработка системы программирования роботов на основе плат Arduino на базе исходных кодов системы программирования КуМир. Для выполнения поставленной цели, требуется выполнить следующие задачи:

- проанализировать исходные коды системы программирования КуМир;
- проанализировать историю развития технологий, используемых для разработки системы программирования;
- доработка и устранение ошибок в исходном коде системы программирования КуМир;
- проанализировать системы программирования плат Arduino и их аналоги;
- разработать отдельный клиент для разработки роботов;
- обновить используемые библиотеки и фреймворки для разработки среды программирования;
- разработать плагин для взаимодействия с платами Arduino.

1 Предметная область. Подходы, понятия, средства

1.1 Среда исполнения и язык программирования КуМир

КуМир (Комплект Учебных МИРов) - система программирования, предназначенная для поддержки начальных курсов информатики и программирования в средней и высшей школе с открытым исходным кодом [1]. Спектр возможностей применения данной системы программирования ограничен, ряд стандартных команд позволяет разрабатывать небольшие приложения с целью изучения основ программирования на процедурных языках.

На данный момент язык КуМир - это язык, с которого хорошо начать, чтобы освоить основы алгоритмического подхода и процедурный стиль программирования. Система КуМир [2]в современном ее состоянии (с подсистемой ПиктоМир) состоит из расширяемого набора исполнителей (или роботов), набора систем программирования и вспомогательных утилит и программ. Расширяемый набор исполнителей (роботов) представляет собой отдельные самостоятельные программы и (или) электронно-механические устройства, имеющие собственное пультовое управление (интерфейс), а также возможность локального или сетевого управления из выполняющей системы. Набор систем программирования:

- система программирования КуМир на школьном алгоритмическом языке, состоящая из редактора-компилятора алгоритмического языка с многооконным интерфейсом, интегрированная с выполняющей системой. Система не является полноценным интерпретатором или компилятором школьного алгоритмического языка. Зачастую, программа на школьном алгоритмическом языке компилируется в некий промежуточный код (подобный подход был использован в языке Java), который затем интерпретируется и исполняется с автоматической генерацией точек останова по событи-

ям и шагам (при пошаговом выполнении). В системе программирования КуМир также отсутствует отладчик в его классическом понимании. Это имеет свои причины, так как сильно упрощает процесс освоения и написания системы программирования КуМир, а также отладку программ. Все изменения используемых в школьной программе величин автоматически визуализируются на полях программы. Визуализируются и результаты логических операций. Кроме того, к системе можно подключать любого исполнителя (робота) из набора и программно управлять им;

- система бестекстового, пиктографического программирования ПиктоМир. В ней учащийся собирает программу из команд исполнителя (робота). Управляющие конструкции языка представлены определенной параметрической организацией алгоритмов. Созданная учащимися программа при выполнении передает команды робота и получает его состояние;
- система программирования на производственном языке программирования C++, Python и т.п [3];

1.2 Архитектура системы программирования КуМир

Помимо инструментов разработки, среда обладает рядом функциональных возможностей - просмотр документации, создание и выполнение практикумов, работа с окном в свернутом режиме и пр. Стартовое окно системы программирования КуМир представлено на рисунке 1.

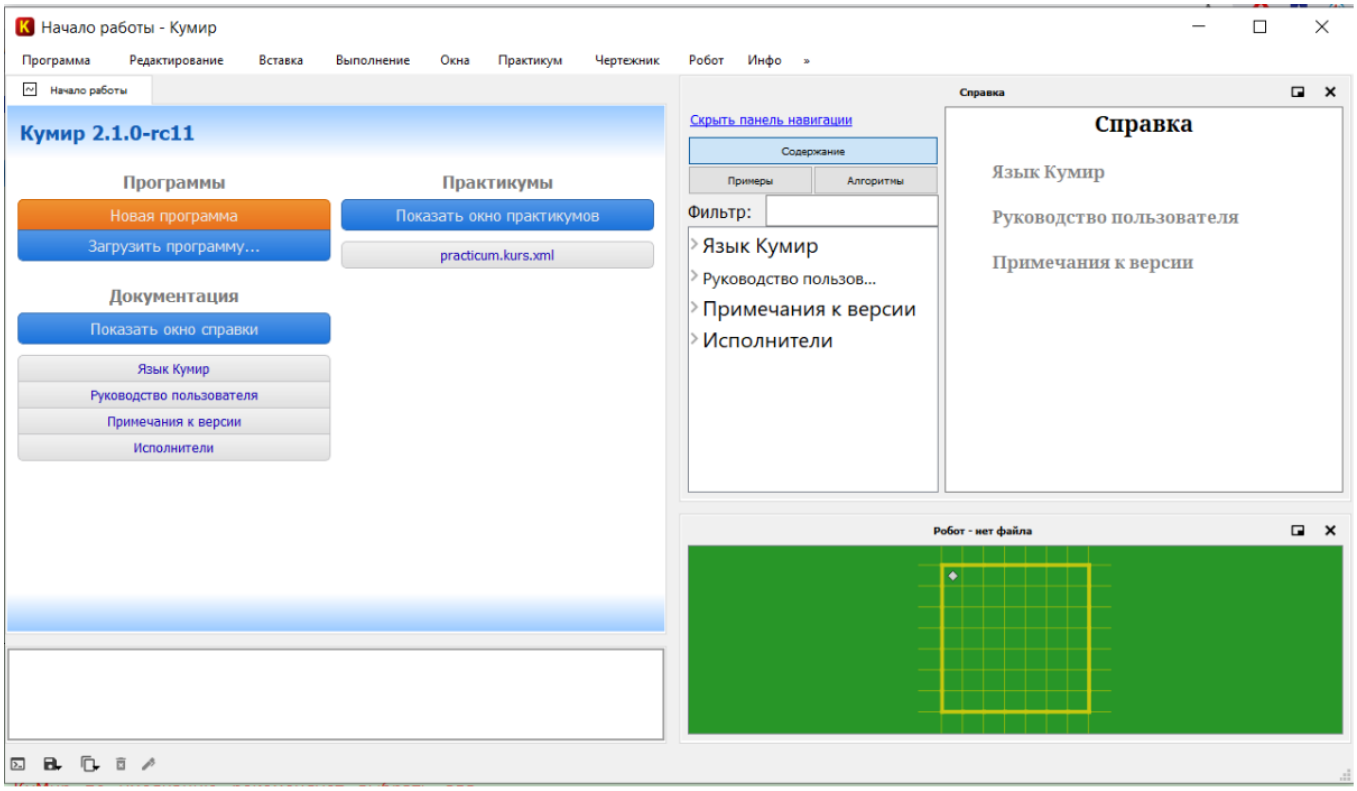

Рисунок 1 — Внешний вид среды программирования КуМир

С точки зрения кода, система программирования КуМир представляет собой монолитное приложение. Исходные коды разделены на 2 основные части:

- библиотеки;
- плагины.

Существует ряд клиентов среды программирования КуМир - для учителя, для ученика старших классов, стандартный клиент. Помимо клиентов, существуют отдельные приложения для компиляции и исполнения кода на языке программирования КуМир. Описанные выше приложения создаются при помощи разделяемого программного кода и вспомогательных инструментов.

К вспомогательным инструментам стоит отнести и ряд функций, упрощающих работу с системой сборки CMake [4]. В исходных кодах можно найти функции упрощающие поиск библиотек для фреймворка QT, определения и установки файлов ресурсов при сборке, а также функции для сборки артефактов приложения(клиентов) из исходных кодов.

Для создания нового артефакта - клиента или приложения командной строки используется CMake-функция $kumir2\_add\_launcher$. По созданному конфигурационному файлу CmakeLists.txt с описанием нового клиента по аналогии с имеющимися, во время сборки приложения, программа-сборщик обнаружит файл конфигурации и с его помощью создаст клиент. Пример конфигурационного файла CmakeLists.txt приведен в листинге 1:

```
project(kumir2-robots)
cmake_minimum_required(VERSION 3.0)

find_package(Kumir2 REQUIRED)

kumir2_add_launcher(
   NAME            kumir2-robots
   SPLASHSCREEN    "splashscreen-classic.png"
  WINDOW_ICON     "window-icon-classic.png"
    APP_ICON_NAME   "kumir2-classic"
   X_ICONS_DIR     "../../../app_icons/linux/hicolor"
    WIN_ICONS_DIR   "../../../app_icons/win32"
   X_NAME          "Kumir Robots Edition"
    X_NAME_ru       "Кумир для роботов"
    X_CATEGORIES    "Education,X-KDE-Edu-Misc"
    CONFIGURATION
  "CourseManager,Editor,ActorArduino,
 ArduinoCodeGenerator\(preload=Files\),
 KumirAnalizer\(preload=Files\),*CodeGenerator,
 KumirCodeRun(nobreakpoints),!CoreGUI\(nosessions\)")
```

Листинг 1 — Пример содержимого файла CMakeLists.txt для нового клиента системы программирования КуМир

Функция $kumir2\_add\_launcher$ позволяет декларативно настроить результат

сборки - указать его название, иконку, а также ряд зависимостей для сборки приложения.

Клиент среды программирования состоит из плагинов и исполнителей. Плагины ссылаются на библиотеки КуМир-а, предоставляя конкретные реализации на основе контрактов [5], описываемых моделями библиотек. Среди инструментов разработчика существуют скрипты для кодогенерации оснастки исполнителей, развертывания приложения в разных операционных системах и генерации CMake-скриптов. Библиотеки представляют собой набор базовых сущностей, реализующих определенную часть функционала. В подавляющем большинстве, классы, описывающие область знания не содержат в себе логики работы с данной областью и представляют лишь анемичные модели [6], [7]для хранения состояния. Для работы с моделями используются генераторы или фабрики. Среди библиотек существует собственная реализация AST-дерева [8]. Библиотека содержит модели, описывающие выражения, типы, модули, алгоритмы, переменные и лексемы. Программный код, использующий данные модели представляет собой плагины для кодогенерации.

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 Оффициальный сайт КуМир [Электронный ресурс]. — 2022. — URL: https://www.niisi.ru/kumir/. Загл. с экр. Яз. рус.

2 Леонов, А. Г. Методика преподавания основ алгоритмизации на базе системы «КуМир» [Электронный ресурс]. — 2009. — URL: https://inf.1sept.ru/view_article.php?ID=200901701. Загл. с экр. Яз. рус.

3 Кушниренко, А. Г. докл. ПиктоМир: пропедевтика алгоритмического языка (опыт обучения программированию старших дошкольников) // Большой московский семинар по методике раннего обуч. информатике. — М.: ИТОРОИ, 2012.

4 Манаев Р.Г. ТЕХНОЛОГИЯ ВНЕДРЕНИЯ НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ В КРУПНЫХ ВЫСОКОНАГРУЖЕННЫХ СИСТЕМАХ С МИНИМИЗАЦИЕЙ ОШИБОК И ВРЕМЕННЫХ ПОТЕРЬ СО СТОРОНЫ РАЗРАБОТЧИКОВ // Инновации и инвестиции. 2020. №12.

5 Маклафлин Б., Поллайс Г., Уэст Д. Объектно-ориентированнный анализ и проектирование. - СПб.: Питер, 2013. - 608с.: ил.

6 Фаулер, Мартин. Шаблоны корпоративных приложений. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2016. - 544с.: ил. - Парал. тит. англ.

7 Эванс, Эрик. Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем.: Пер. с англ. - СПб.: ООО “Диалектика”, 2020 - 448с.: ил. - Парал. тит. англ.

8 Ахо, Альфред В., лам, Моника С., Сети, Рави, Ульман, Джеффри Д. Компиляторы: принципы, технологии и инструментарий. 2-е изд.: Пер. с англ. - М.: ООО “И.Д. Вильямс”, 2018 - 1184с.: ил. - Парал. тит. англ.

9 Пронин А.А., Синельников Е.А. Модули в языке программирования КуМир 2.0 // Информационные технологии в образовании: сборник / редакционная коллегия: С. Г. Григорьев [и др.]. – Саратов: Саратовский университет [издание], 2022. – Вып. 5: материалы XIV Всероссийской научно-практической конференции «Информационные технологии в образовании» (ИТО-Саратов-2022), Саратов, 28-29 октября 2022 г. – 290 с. : ил. (9,19Мб). – URL: https://sgu.ru/node/197426. – Режим доступа: Свободный. Продолжающиеся издания СГУ на сайте www.sgu.ru. [207-211]

10 Смирнов Максим докл. Модернизация унаследованных приложений // конференция ArchDays 2023

11 Исходный код среды исполнения КуМир [Электронный ресурс]. — URL: https://github.com/a-a-maly/kumir2 Загл. с экр. Яз. рус

12 Кушниренко, А. Г. Опыт интеграции цифровой образовательной среды КуМир в платформу Мирера // Объединенная конференция "СПО: от обучения до разработки": материалы конференции / Под ред. В. Л. Чёрный. — МАКС Пресс, 2022. — С. 24–30.

13 Список версий фреймворка QT [Электронный ресурс]. — URL: https://wiki.qt.io/Portal:Quick_Access

14 Вареница Виталий Викторович, Марков Алексей Сергеевич, Савченко Владислав Вадимович, Цирлов Валентин Леонидович ПРАКТИЧЕСКИЕ АСПЕКТЫ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ ПРИ ПРОВЕДЕНИИ СЕРТИФИКАЦИОННЫХ ИСПЫТАНИЙ ПРОГРАММНЫХ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ // Вопросы кибербезопасности. 2021. №5 (45).

15 Амини Камран. Экстремальный Си. Параллелизм, ООП и продвинутые возможности. — СПб.: Питер, 2021. — 752 с.: ил. — (Серия «Для профессионалов»).

16 Бобков В.А., Черкашин А.С. Обработка и визуализация пространственных данных на гибридном вычислительном кластере // Прикладная информатика. 2014. №4 (52).

17 Шабалин, К. В. Формирование креативных способностей школьников при выполнении проектов на базе платформы Arduino / К. В. Шабалин // Педагогическое образование в России. — 2022. — No 2. — С. 135–140.

18 Глазов Сергей Юрьевич, Сергеев Алексей Николаевич, Усольцев Вадим Леонидович ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ ПЛАТФОРМЫ ARDUINO В УЧЕБНОМ ПРОЦЕССЕ ПЕДАГОГИЧЕСКОГО ВУЗА И ОБЩЕОБРАЗОВАТЕЛЬНЫХ ШКОЛ // Известия ВГПУ. 2021. №10 (163).

19 Серёгин, М. С. Использование платформы Arduino в образовательной деятельности / М. С. Серёгин // Инновационная наука. — 2019. — No 6. —С. 62–64.

20 Серёгин, М. С. Использование платформы Arduino в образовательной деятельности / М. С. Серёгин // Инновационная наука. — 2019. — No 6. — С. 62–64.

21 Исходный код приложения Arduino IDE первой версии [Электронный ресурс]. — URL: https://github.com/arduino/Arduino

22 Исходный код приложения Arduino IDE современной версии [Электронный ресурс]. — URL: https://github.com/arduino/arduino-ide

23 Борис Черный. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений. - СПб.: Питер, 2021. - 352 с.: ил. - (Серия “Бестселлеры O’Reily”)

24 Мардан Азат. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. — СПб.: Питер, 2019. — 560 с.: ил. — (Серия «Библиотека про-

граммиста»).

25 Исходный код приложения Arduino CLI [Электронный ресурс]. — URL: https://github.com/arduino/arduino-cli

26 Документация к приложению Arduino CLI [Электронный ресурс]. — URL: https://arduino.github.io/arduino-cli/0.35/

27 Оффициальный сайт протокола grpc [Электронный ресурс]. — URL: https://grpc.io/

28 Индрасири Касун, Курупу Данеш. gRPC: запуск и эксплуатация облачных приложений. Go и Java для Docker и Kubernetes. - СПб.: Питер, 2021. - 224с.: ил. - (Серия “Бестселлеры O’Reily”)

ПРИЛОЖЕНИЕ А

Исходный код dockerfile, унифицирующий настройки среды запуска системы тестирования

```
FROM ubuntu

ARG path_to_tests_folder="./Tests"

LABEL EMAIL=gorka19800@gmail.com

#install all the necessary libs and apps
RUN apt-get update
RUN apt-get dist-upgrade -y
RUN echo "8"| apt-get install -y qttools5-dev-tools
RUN apt-get install -y git python3 cmake qtbase5-dev g++ libqt5svg5-dev
 libqt5x11extras5-dev qtscript5-dev libboost-system-dev zlib1g zlib1g-dev
#setup git
RUN git config --global user.email "gorka19800@gmail.com"
RUN git config --global user.name "Test suit"
#clone repo and prepare for building kumir-to-arduino translator
RUN mkdir /home/Sources
WORKDIR /home/Sources/
RUN git clone https://github.com/CaMoCBaJL/kumir2
WORKDIR kumir2/
RUN git pull
RUN git checkout translator_tests
RUN git merge -s ours --no-edit origin/ArduinoFixes
RUN mkdir build
WORKDIR build/
#build translator
RUN cmake -DUSE_QT=5 -DCMAKE_BUILD_TYPE=Release ..
RUN make -j 18
#start tests
```

```
30 WORKDIR ../kumir_tests/
31 RUN touch test_results.log
32 RUN python3 test_script.py -d -o ./test_results.log
33 -tr ../build/bin/kumir2-arduino -t \$path_to_tests_folder
```

ПРИЛОЖЕНИЕ Б

Исходный код скрипта, производящего тестирование транслятора

```
import os
import sys
import subprocess

#constants
class CONSOLE_BG_COLORS:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKCYAN = '\033[96m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'

class TEST_RESULT_STATE:
    COMPLETED = 1
    MISSING_EXPECTATION = 2
    COMPILER_ERROR_HAPPEND = 3
    FAILED = -1
    NONE = 0

COMPILER_ERROR_LABEL = "ERROR!"

EXPECTATION_FILE_EXTENTION = ".exp"
SOURCE_FILE_EXTENTION = ".kum"

CONTROL_CHARACTERS = ["\n", "\r", "\t", " "]
BYTES_TO_READ_COUNT = 1024
```

```
CHAR_THRESHOLD = 0.3
TEXT_CHARACTERS = ''.join(
    [chr(code) for code in range(32, 127)] +
    list('\b\f\n\r\t')
)
BINARY_CHAR_EXAMPLE = '\x00'

TESTS_FOLDER_NAME = "Tests"

ARGS = {"help": ["-h", "--help"],
        "translator": ["-tr", "--translator"],
        "output": ["-o", "--output"],
        "duplicate": ["-d", "--duplicate"],
        "skip-successfull": ["-ss", "--skip-successfull"],
        "skip-failed": ["-sf", "--skip-failed"],
        "skip-without-expectation": ["-swe", "--skip-without-expectation"],
        "skip-with-compiler-error": ["-sce", "--skip-with-compiler-error"],
        "brief": ["-b", "--brief"],
        "path-to-tests": ["-t", "path-to-tests"]
        }

ERRORS = {
    "wrong input": "Wrong args input - you should type path to file after -c or
     -o flags!",
    "file not exist": "File doesn't exist!",
    "wrong file extention": "File extention for translator is not correct! It should
     be a bin-file.",
    "no path to tests": "Wrong args input - you should type path to tests folder
     after -t flag!"
}

#data structure to store test results
class TestResult:
```

```
64
65      __text_color = CONSOLE_BG_COLORS.OKGREEN
66      __header_text = ""
67      SKIPPED_TEST_TYPES = [TEST_RESULT_STATE.NONE]
68
69      def __init__(self, name: str, source: str, expectation: str, result: str):
70          self.name = name
71          self.source_file_name = source
72          self.expectation_file_name = expectation
73          self.resultFileName = result
74          self.state = TEST_RESULT_STATE.NONE
75
76      def __str__(self):
77          self.__setup_output()
78          additional_test_data = f'''
79          Test group expectations: {self.expectation_file_name}
80          Test group results: {self.resultFileName}.
81          To get more detail info about comparison results, print:
82          vimdiff {self.source_file_name} {self.expectation_file_name}
83           {self.resultFileName}
84          or vimdiff {self.expectation_file_name} {self.resultFileName}
85          {CONSOLE_BG_COLORS.ENDC}
86          '''
87          return f'''{self.__text_color}
88          Test {self.name}.
89          {self.__header_text}
90          Sources for the test: {self.source_file_name}
91          {additional_test_data if self.state is TEST_RESULT_STATE.COMPLETED
92           or self.state is TEST_RESULT_STATE.FAILED else ""}
93          '''
94
95      def __setup_output(self):
96          if self.state == TEST_RESULT_STATE.COMPLETED:
```

```
97              self.__text_color = CONSOLE_BG_COLORS.OKGREEN
98              self.__header_text = "Congratulations! Test completed successfully!"
99          elif self.state == TEST_RESULT_STATE.MISSING_EXPECTATION:
100             self.__text_color = CONSOLE_BG_COLORS.OKCYAN
101             self.__header_text = "Oh! Test didn't complete: no expectation"
102         elif self.state == TEST_RESULT_STATE.COMPILER_ERROR_HAPPEND:
103             self.__text_color = CONSOLE_BG_COLORS.WARNING
104             self.__header_text = "Oh! Test didn't complete: compiler error"
105         elif self.state == TEST_RESULT_STATE.FAILED:
106             self.__text_color = CONSOLE_BG_COLORS.FAIL
107             self.__header_text = "Sorry, but test failed..."
108
109 class TestSection:
110
111     def __init__(self, section_name) -> None:
112         self.name = section_name
113         self.test_results = []
114
115     def __str__(self) -> str:
116         columns, _ = os.get_terminal_size()
117         if (len(self.test_results) > 0):
118             return f"""
119             {CONSOLE_BG_COLORS.WARNING + "-" *columns + CONSOLE_BG_COLORS.ENDC}
120             Test section {self.name} starts here:
121             {os.linesep.join(list(map(lambda test_result: str(test_result),
122              self.test_results)))}
123             End of {self.name} section tests
124             {CONSOLE_BG_COLORS.WARNING + "-" *columns + CONSOLE_BG_COLORS.ENDC}
125             """
126
127         return ''
128
129 #functions
```

```
130 def remove_control_characters(data_array):
131     result = []
132     for i in data_array:
133         for cc in CONTROL_CHARACTERS:
134             i = i.replace(cc, "")
135 
136         if i:
137             result.append(i)
138 
139     return result
140 
141 def get_file_data(filename):
142     if not os.path.exists(filename):
143         return ''
144     file = open(filename, "r")
145     result = file.readlines()
146     file.close()
147     return result
148 
149 def has_errors(text):
150     return COMPILER_ERROR_LABEL in text
151 
152 def process_sources(source_filename, path_to_translator):
153     path, _ = os.path.splitext(source_filename)
154     result_filename = path + ".kumir.c"
155 
156     #call kumir2-arduino with params:
157     --out="path_to_cwd/results/test_name.kumir.c" -s ./test_name.kum
158     popen = subprocess.Popen([path_to_translator,
159                               f'--out={result_filename}',
160                               '-s',
161                               source_filename],
162                              stdout=subprocess.PIPE)
```

```
163     popen.wait()
164     popen.stdout.read()
165
166     return result_filename
167
168 def compare_data(expected_data, processed_data) -> TEST_RESULT_STATE:
169     result_without_kumir_ref = remove_control_characters(processed_data[2:])
170     expected_data = remove_control_characters(expected_data)
171
172     if (has_errors(result_without_kumir_ref)):
173         return TEST_RESULT_STATE.COMPILER_ERROR_HAPPEND
174
175     for i in range(len(result_without_kumir_ref)):
176         if result_without_kumir_ref[i] != expected_data[i]:
177             return TEST_RESULT_STATE.FAILED
178
179     return TEST_RESULT_STATE.COMPLETED
180
181 def get_test_result_type_counters(test_sections):
182     counters = {
183         TEST_RESULT_STATE.COMPLETED: 0,
184         TEST_RESULT_STATE.FAILED: 0,
185         TEST_RESULT_STATE.COMPILER_ERROR_HAPPEND: 0,
186         TEST_RESULT_STATE.MISSING_EXPECTATION: 0,
187     }
188
189     for test_section in test_sections:
190         for test_result in test_section.test_results:
191             counters[test_result.state] += 1
192
193
194     return counters
195
```

```
196 #log comparison results to file
197 def log_test_results(logs_filename, data_to_log: TestSection, log_to_console):
198     result_type_counters = get_test_result_type_counters(data_to_log)
199     completed_tests_count, failed_tests_count,
200      compiler_error_happend_tests_count,
201      missing_expectations_tests_count = \
202     [result_type_counters[k] for k in result_type_counters]
203 
204     log_data = []
205     for test_section in data_to_log:
206         test_section.test_results = list(filter
207         (lambda test_result: test_result.state not in
208         TestResult.SKIPPED_TEST_TYPES, test_section.test_results))
209         log_data.append(test_section)
210 
211     log_data_strings = list(map(lambda x: str(x), log_data))
212     log_data_strings.insert(0, f'''
213     There were found: {len(log_data)} tests.
214     {CONSOLE_BG_COLORS.OKGREEN} Completed:
215     {completed_tests_count} {CONSOLE_BG_COLORS.ENDC}
216     {CONSOLE_BG_COLORS.FAIL} Failed:
217     {failed_tests_count} {CONSOLE_BG_COLORS.ENDC}
218     {CONSOLE_BG_COLORS.WARNING} With compiler error happend:
219      {compiler_error_happend_tests_count} {CONSOLE_BG_COLORS.ENDC}
220     {CONSOLE_BG_COLORS.OKCYAN} Missed expectation file:
221     {missing_expectations_tests_count} {CONSOLE_BG_COLORS.ENDC}''')
222 
223     if not os.path.exists(logs_filename):
224         open(logs_filename, "a").close()
225 
226     log_file = open(logs_filename, "a")
227     log_file.writelines(log_data_strings)
228     log_file.close()
```

```
229
230     if log_to_console:
231         print(f"{os.linesep}".join(log_data_strings))
232
233 def is_binary_file(filename):
234     file_stream = open(filename, 'rb')
235     file_content = file_stream.read(BYTES_TO_READ_COUNT)
236     file_stream.close()
237
238     if not len(file_content):
239         #file is empty, nothing to read
240         return False
241
242     if ord(BINARY_CHAR_EXAMPLE) in file_content:
243         #file contains binary symbols
244         return True
245
246     binary_chars = file_content.translate(TEXT_CHARACTERS)
247     return float(len(binary_chars)) / len(file_content) > CHAR_THRESHOLD
248
249 def validate_file_name(filename: str, is_log_file: bool):
250     if not os.path.exists(filename):
251         print(filename + ERRORS.get("file not exist"))
252         sys.exit(2)
253     if not os.path.isfile(filename):
254         print(ERRORS.get("wrong input"))
255         sys.exit(2)
256     if not is_log_file and not is_binary_file(filename):
257         print(ERRORS.get("wrong file extention"))
258         sys.exit(2)
259
260 def show_help():
261     print(""" kumir2-arduino tester.
```

```
Description:
Approach of this app is to debug the work of kumir2
to arduino translator. It uses compiled translator's
instance, pre-builded locally on PC.
To start the work you should input path to compiler
and path to logs file.
Flags:
[-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.

```

```
295     [-b] [--brief] - skip open log info about all tests.
296     """)
297
298 def process_args():
299     result = ["", "", False, ""]
300     if ARGS["help"][0] in sys.argv or ARGS["help"][1]
301     in sys.argv:
302         show_help()
303         sys.exit(2)
304
305     if ARGS["skip-successfull"][0] in sys.argv or ARGS["
306     skip-successfull"][1] in sys.argv:
307         TestResult.SKIPPED_TEST_TYPES.append(
308         TEST_RESULT_STATE.COMPLETED)
309
310     if ARGS["skip-without-expectation"][0] in sys.argv
311     or ARGS["skip-without-expectation"][1] in sys.argv:
312         TestResult.SKIPPED_TEST_TYPES.append(TEST_RESULT_STATE
313         .MISSING_EXPECTATION)
314
315     if ARGS["skip-with-compiler-error"][0] in sys.argv or
316      ARGS["skip-with-compiler-error"][1] in sys.argv:
317         TestResult.SKIPPED_TEST_TYPES.append(TEST_RESULT
318         _STATE.COMPILER_ERROR_HAPPEND)
319
320     if ARGS["skip-failed"][0] in sys.argv or ARGS[
321     "skip-failed"][1] in sys.argv:
322         TestResult.SKIPPED_TEST_TYPES.append(
323         TEST_RESULT_STATE.FAILED)
324
325     if ARGS["brief"][0] in sys.argv or ARGS["brief"][1] in sys.argv:
326         TestResult.SKIPPED_TEST_TYPES = [
327             TEST_RESULT_STATE.COMPILER_ERROR_HAPPEND,
```

```
328             TEST_RESULT_STATE.COMPLETED,
329             TEST_RESULT_STATE.FAILED,
330             TEST_RESULT_STATE.MISSING_EXPECTATION
331             ]
332
333     args = sys.argv[1:]
334     for i in range(1, len(args)):
335         if args[i - 1] in ARGS["translator"] or args[i - 1]
336          in ARGS["output"] or args[i - 1] in ARGS
337          ["path-to-tests"]:
338             if (os.path.isfile(args[i])):
339                 validate_file_name(args[i], False if args[i - 1]
340                  in ARGS["translator"] else True)
341
342             if args[i - 1] in ARGS["translator"]:
343                 result[0] = os.path.abspath(args[i])
344             elif args[i - 1] in ARGS["output"]:
345                 result[1] = os.path.abspath(args[i])
346             elif args[i - 1] in ARGS["path-to-tests"]:
347                 result[3] = os.path.abspath(args[i])
348         elif args[i-1] in ARGS["duplicate"]:
349             result[2] = True
350
351     return result
352
353 def get_files_with_absolute_paths(folder_name):
354     path_to_folder = os.path.join(os.getcwd(), folder_name)
355     files =
356     list(map(lambda x: os.path.join(path_to_folder, x),
357      os.listdir(path=path_to_folder)))
358     files.sort()
359
360     return files
```

```
361
362 def get_source_and_expectation(dir_files):
363     sources = []
364     expectations = []
365     for file in dir_files:
366         if (os.path.isfile(file)):
367             ext = os.path.splitext(file)[1]
368             if (ext == EXPECTATION_FILE_EXTENTION):
369                 expectations.append(file)
370             elif (ext == SOURCE_FILE_EXTENTION):
371                 sources.append(file)
372
373     return [sources, expectations]
374
375 def get_folder_contents_full_paths(path_to_folder):
376     return list(
377         map(
378             lambda x: os.path.join(os.sep, path_to_folder, x),
379             os.listdir(path_to_folder)
380             )
381         )
382
383 def calculate_test_sections(path_to_tests_folder):
384     result = []
385     test_folder_paths = get_folder_contents_full
386     _paths(path_to_tests_folder)
387     for test_folder_path in test_folder_paths:
388         if os.path.isfile(test_folder_path):
389             continue
390
391         result.append(TestSection(test_folder_path.
392         split(os.sep)[-1]))
393         test_paths = get_folder_contents_full_paths
```

```
394         (test_folder_path)
395         for test_dir_path in test_paths:
396             source_files, expectation_files =
397           get_source_and_expectation(get_folder_contents_full_paths
398             (test_dir_path))
399             if os.path.isfile(test_dir_path) or len
400             (expectation_files) > 1 or len(source_files)
401              < 1:
402                 continue
403
404             result[-1].test_results.append(TestResult(
405                 test_dir_path.split(os.sep)[-1],
406                 source_files[0],
407                 "", ""
408                 )
409             )
410
411             if len(expectation_files) == 1:
412                 result[-1].test_results[-1]
413           .expectation_file_name  = expectation_files[0]
414                 result[-1].test_results[-1].resultFileName
415                  = process_sources(source_files[0],
416                   args_data[0])
417                 result_data = get_file_data(result[-1]
418
419                 .test_results[-1].resultFileName)
420                 expected_data =
421       get_file_data(result[-1].test_results[-1].expectation_file_name)
422                 result[-1].test_results[-1].state
423                 = compare_data(expected_data, result_data)
424             else:
425                 result[-1].test_results[-1].state =
426                 TEST_RESULT_STATE.MISSING_EXPECTATION
```

```
427
428     return result
429
430 if __name__=="__main__":
431     args_data = process_args()
432
433     if not args_data[3]:
434         print("Didn't find any test to execute. Shutting down.")
435         sys.exit()
436
437     test_results = calculate_test_sections(args_data[3])
438
439     log_test_results(args_data[1], test_results, args_data[2])
```