окрашенные в синий цвет комплексы с ионами калия, натрия, а также цезия и рубидия. Установлено, что интенсивность окраски раствора зависит от природы катиона, его концентрации, а также длительности реакции (при постоянной концентрации щелочи). При высокой концентрации щелочей (1М) окраска теряет свою интенсивность, что может свидетельствовать о неустойчивости комплекса. Зависимости оптической плотности исследуемого раствора от концентраций KOH и NaOH показаны на рис. 1, от продолжительности реакции – на рис. 2.

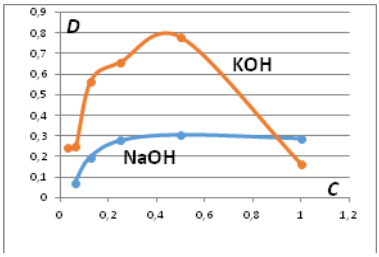

Рисунок 1 - Зависимость оптической плотности раствора от концентраций KOH и NaOH

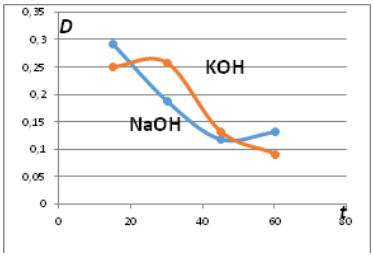

Рисунок 2 - Зависимость оптической плотности раствора от длительности проведения реакции

Таким образом, показано, что реакция щелочного гидролиза 5,6-диметил-3-(2'-метилбензимидазолил -1')-1,2,4-триазина может быть использована для спектрофотометрического определения катионов щелочных металлов, 5,6-диметил-3-(2'-метилимидазолил -1')-1,2,4-триазин в аналогичных условиях окрашенных соединений не образует.

УДК 004.054

ОПЫТ СОЗДАНИЯ БИБЛИОТЕКИ ИНСТРУМЕНТОВ ДЛЯ ТЕСТИРОВАНИЯ КОНСОЛЬНЫХ ПРИЛОЖЕНИЙ НА ЯЗЫКАХ ПРОГРАММИРОВАНИЯ C# И C++

В. Д. Кротков[1]

Научный руководитель: А. Н. Даниленко, к.т.н., доцент

Ключевые слова: библиотека инструментов, языки программирования C# и C++, тестирование приложений

В настоящее время для тестирования сложных систем и проверки их архитектуры на корректность используются отдельные инструменты отладки кода, Clang Power Tools, JUnit [1]. Однако не всегда можно легко и быстро проверить, правильно ли работает алгоритм, протестировать реакцию приложения или программного

[1] Владимир Денисович Кротков, студент группы 6313-020302D,
email: galaktikaonline@yandex.ru

компонента на действия пользователя. В настоящей работе приведено описание функционала библиотеки, предназначение которой – упрощение создания консольных приложений на языках C# и C++ путем предоставления программисту возможностей использовать встроенное настраиваемое меню произвольной степени вложенности, специальные средства для осуществления контроля над переменными программы и отслеживания состояния программных сущностей [2].

Предлагаемая авторами библиотека дает программисту следующие возможности:

– использовать встроенное меню;

– создавать пул рабочих переменных;

– автоматически генерировать программные данные;

– проводить контролируемые тесты нужного блока кода;

– получать результаты тестирования;

– получать информацию о runtime-ошибках.

Меню – главный элемент библиотеки. При запуске оно позволяет выбрать на выполнение один из своих пунктов, представленных в виде сигнатур функций или лямбда-функций [3]. У меню присутствуют три режима работы:

– ручной – программист сам выбирает, какой пункт меню запустить и какие действия произвести в пределах этого пункта;

– полуавтоматический – программист вводит последовательность команд, которые нужно выполнить, а меню перехватывает управление консолью и вводит в нее эти команды;

– автоматический – меню само генерирует последовательность команд и необходимые данные и затем запускает нужные пункты.

В данной библиотеке меню представлено классом Menu, а пункты меню – базовым классом MenuItem, причем Menu содержит список объектов MenuItem и само наследуется от MenuItem. Список пунктов меню имеет древовидную структуру, ветками которой служат объекты Menu, а листьями – объекты CommonMenuItem. Был применен структурный шаблон проектирования Composite. В такой архитектуре оставлена возможность расширения иерархии наследования, строящейся вокруг MenuItem, без изменения кода, взаимодействующего с пунктами меню.

Так как конструирование объекта класса Menu может оказаться громоздким, было принято архитектурное решение – реализовать порождающий шаблон Builder, написав класс MenuBuilder. Его задача – упростить создание сложного объекта класса Menu, разбив это создание на цепочку простых вызовов. Строитель MenuBuilder тем отличается от других строителей, что он позволяет одновременно создавать объекты класса Menu непосредственно на месте пунктов текущего меню. Такая возможность реализована за счет перевода

текущего строящегося меню в стек и постановки на строительство нового меню. По окончанию конструирования вершина стека удалится из него и станет текущим конструирующимся меню.

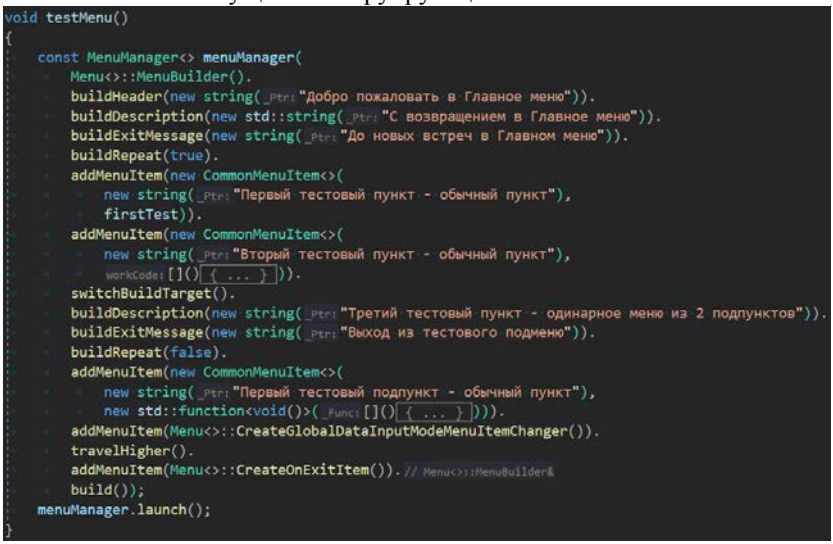

Рисунок 1 – Процесс конструирования меню на клиентской стороне

На текущем этапе разработки библиотеки значительное место занимает концепт контролируемых тестов [4]. Он предполагает то, что автоматика должна уберегать программиста от ошибки при вводе тестовых данных, чтобы он мог быстро, надежно и удобно тестировать необходимые блоки кода. Поэтому все меню и другие элементы библиотеки находится под надзором системы контроля ввода данных.

Программист может использовать встроенные проверки на корректность или сам определять, какой именно ввод данных считать правильным. Для этого в библиотеке есть функционал ввода строки по заранее заданному шаблону. Далее программист может использовать полученную строку для построения собственного объекта. Однако такого функционала не всегда достаточно. Чтобы завершить концепт контролируемых тестов и целиком переложить получение корректных данных на автоматику, был написан шаблонный класс Parser, задача которого – непосредственно получать объекты нужных типов из введенной строки. В числе преимуществ такого подхода – более гибкая настройка конвертации строк в объекты и возможности дальнейшей настройки классов-конвертеров.

Программист может настроить меню на выполнение серии тестов блока кода. Результаты тестирования он может получать в виде информации в консоли в виде цепочке из всех пунктов, которые запустились на выполнение в меню, и результатов их работы; отчета в текстовом файле. При выполнении цепочки команд меню печатает содержимое консоли в отдельный файл; объекта класса Results.

Все способы получения результатов включают в себя получение информации о количестве успешных тестов, которые прошел блок кода, о том, какие тесты и на каких наборах данных оказались неудачными, об исключениях, сгенерировавшихся в процессе, и о времени выполнения программы.

При возникновении ошибки в пределах блока меню пользователю будет доступен отчет о месте, времени, характере и заложенном описании ошибки. Ему также придет запрос, прекратить или продолжить выполнение программы.

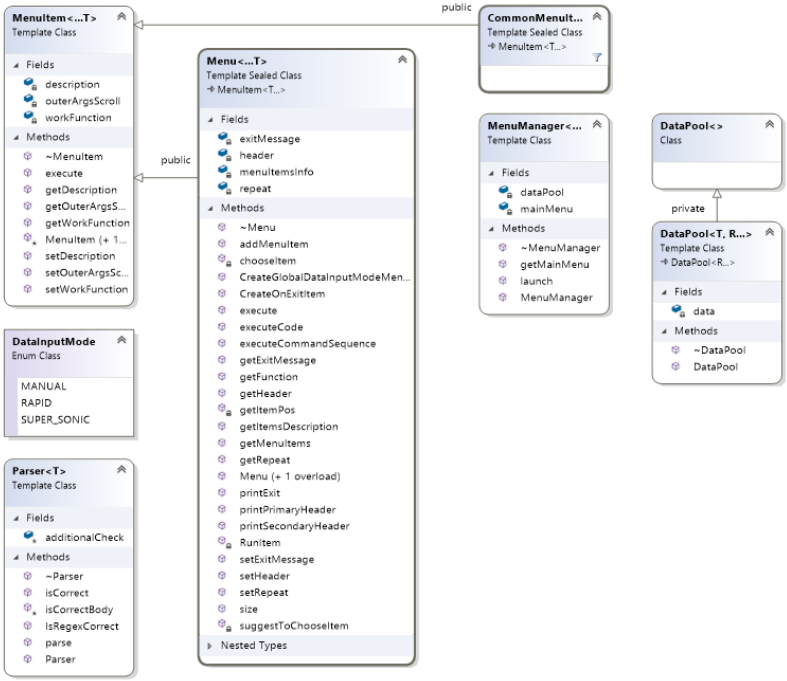

Рисунок 2 – Диаграмма основных взаимосвязей элементов библиотеки и их состава

Основные преимущества разработанной библиотеки:

– существенное снижение временных затрат на написание однотипного кода;

– облегчение задачи тестирования приложений;

– возможность автоматической генерации программных данных;

– возможность получения результатов тестирования с указанием информации о runtime-ошибках.

Библиотека была интегрирована и апробирована на реальных задачах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Freeman E. Head First Design Patterns: a Brain-Friendly Guide / E. Freeman, B. Bates, K. Sierra, E. Robson // ISBN: 978-0596007126. – СПб.: Питер, 2018. – 656 с.

2. Gamma E. Design Patterns Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides // ISBN:978-5-496-03210-0. – Addison-Wesley, 1994. – 395 p.

3. How to write Regular Expressions? // geeksforgeeks.org [Электронный ресурс]. – URL: https://www.geeksforgeeks.org/write-regular-expressions/ (дата обращения: 10.12.2019).

4. Variadic templates. Tuples, unpacking and more // habr.com [Электронный ресурс]. – URL: https://habr.com/ru/post/228031/ (дата обращения: 10.12.2019).

УДК 621.45.01

DETERMINATION OF MATERIAL FOR THE PRODUCTION OF A GAS TURBINE ENGINE

Г. А. Косов[1]

Научный руководитель: Н. А. Слобожанина, к.ф.н., доцент

Ключевые слова: gas turbine engine, determination of material, requirements, main loads

This recommendation is based on the loads occurring in each section of the engine.

The main considered loads occurring in the processes inside the engine are: gas-dynamic, gas-static; mass loads; temperature loads.

In accordance with the different causes of loads, the engine was divided into sections with similar loads and actions with the working body. Then in each section the changes in gas parameters were taken into account.

Therefore, the most suitable materials for each section are:

[1] Герман Андреевич Косов, студент группы 2209-130303D,
email: germankosov@inbox.ru