Структура ВКР

1. Общая часть – проблема, известные пути её решения, наша идея её решения.
   1. Проблема – это АСНИ в области научных исследований и, в частности, исследования стандартов мобильной связи.
      1. Обработка потоков данных, их трансформация, формирование выходных потоков
      2. Проработка возможности формирования контрольных точек для сохранения и возобновления процессов обработки потоков данных
      3. Сохранение параметров, индикаторов и результатов преобразований

Проблема автоматизации и повышения эффективности научных исследований существовала с самого начала их появления. В настоящее время существует множество решений этой проблемы. Одним из них является использование в научных исследованиях автоматизированной системы научных исследований (сокращённо АСНИ), реже используются термины САНИ – система автоматизации научных исследований и САЭ – система автоматизации эксперимента.

Автоматизированная система научных исследований – это программно-аппаратный комплекс, использующий вычислительную технику в своей основе и предназначенный для проведения исследований практически в любой сфере науки. В основе АСНИ лежит идея моделирования объектов и их отношений из реальной жизни и оперирование этими моделями для симуляции изучаемых процессов. Научные исследования в большинстве своём предполагают произведение точных и алгоритмически сложных вычислений. Производимые старым способом, то есть вручную, они занимают много времени и усилий исследователя. Автоматизированные системы повышают эффективность научных исследований за счёт своих вычислительных мощностей, ускоряя проведение вычислений во много раз.

Сложно придумать сферу, где не пригодилась бы АСНИ. Такие системы можно применять в экономике, физике, биологии, геологии, медицине, промышленности и так далее. В частности, АСНИ могут эффективно применяться в области исследования стандартов мобильной связи.

Мобильная связь – это способ связи с беспроводным доступом к каналу связи, то есть без использования кабеля. Вместо этого, в основе мобильной связи лежит принцип радиосвязи – способ передачи сообщений на расстоянии посредством радиоволн. Благодаря ему можно производить соединение между перемещающимися в пространстве абонентами, из-за чего связь и называется мобильной. Исследования в области мобильной связи предполагают большие и сложные эксперименты, симулирующие работу системы с множеством различных объектов, что предполагает обработку объёмных потоков данных, вычисления с множеством параметров и применением различных алгоритмов обработки на одних и тех же этапах для сравнения их эффективности, формирование, хранение и анализ выходных потоков и так далее. Для проведения исследований такого масштаба удобно использовать автоматизированную систему научных исследований.

**Пути решения проблемы – в настоящее время это решается чисто разработкой симуляционных систем. Какие есть АСНИ и что они могут.**

В нынешнее время научных открытий в разных сферах трудится немалое количество ученых, инженеров, конструкторов и других профессий, использующих сложное и дорогое оборудование. Уже из этого факта понятно, что на проведение экспериментов и исследований затрачивается огромное кол-во средств и времени. А точность и эффективность этих исследований зависит от уровня использования компьютерной техники.

И поэтому, для решения данных проблем, а также для решения проблем автоматизации и повышения эффективности исследований (в которую включаются различные сферы науки, включая и мобильную связь), единственный вариант выхода — это только разработка АСНИ, АСУ и других подобных автоматизированных систем. При их всеобщем внедрении и развитии, это позволило бы максимально сократить время и кол-во затраченных ресурсов на исследование, но при это увеличить его точность и эффективность.

Существуют различные виды автоматизированных систем (АС), такие как система управления (АСУ) или ее вариации, связанные с другими процессами (например, АСУТП), система автоматизированного проектирования (САПР) и др., но АСНИ выделяется среди них выходной информацией. В первую очередь это данные с наличием качественной обработкой и уменьшенным итоговым объемом, на основе которых получаются математические модели различных исследований (эксперименты, расчеты и т.д.), процессов и объектов. Кроме этого в АСНИ могут быть использованы и готовые мат. модели для анализа поведения тех или иных объектов исследования или для их уточнения (включая и уточнение самих моделей). Итоги всех операций и исследований могут выводиться в наиболее удобной форме, будь то графически, либо с помощью символов.

На данный момент уже существуют различные АСНИ, которые используются во многих странах мира и помогают проводить исследования.

Один из примеров это EPICS (Экспериментальная физика и система управления производством), разработанная в Лос-Аламосской Национальной Лаборатории (АНЛ), США. Эта АСНИ является набором специальных инструментов, применяемые для разработки и управления распределенными системами в научных и экспериментальных областях физики и промышленности (среди них телескопы, ускорители частиц и другие установки).

Кроме того, EPICS предоставляет широкий спектр возможностей для работы с данными, включая использование SCADA (диспетчерское управление и сбор данных).

Для взаимодействия между компьютерами используется технологии клиент–сервер и публикации–подписки. Серверы или “контроллеры ввода-вывода” (IOCS), которые в режиме реального времени собирают разнообразные данные с подключенных к ним измерительных устройств. Затем эти данные предоставляются клиентам с помощью широкополосного доступа к каналам (CA) или благодаря сетевым протоколам pvAccess, которые специально разработаны для приложений, работающих в реальном времени, таких как эксперименты в научной сфере.

В архитектуре системы присутствуют IOCs (Индикаторы компрометации), которые отвечают за безопасность и имеют базу данных "записей", указывающие на возможное наличие уже существующей угрозы и обеспечивает важной информацией после утечки данных или любого другого нарушения. Эти индикаторы могут быть размещены на стандартных серверах, ПК и на других стандартных процессорах для встраиваемых систем (такие как VME).

Информация, сохраненная в каждой записи IOCs, представлена уникальными идентификаторами, которые известны как переменные процесса (PV). Они доступны через сетевые каналы, обеспечиваемые протоколом CA или pvAccess, что позволяет эффективно и безопасно взаимодействовать между различными системами и устройствами.

EPICS предлагает разнообразие в спектре типов записей для различных операций ввода и вывода (среди которых аналоговый или двоичный форматы) обеспечивает различным функционалом, таким как проведение вычислений. Пользователь имеет возможность создавать свои типы записей, настраивая их под свои потребности и нужды. Каждая запись включает в себя набор полей, содержащих как статическую, так и динамическую информацию, определяющую при этом поведение локальных или удаленных запросов различных функций. Описание большинства типов записей описаны в справочном руководстве EPICS.

В дополнение у нее есть в наличии пакеты графического интерфейса пользователя, которые предоставляют необходимую информацию и возможность взаимодействия с фотоэлектрическими данными с помощью стандартных виджетов дисплея, таких как цифровые индикаторы и текстовые поля. В качестве примеров можно привести EDM, MEDM и CSS, предоставляющие обширные возможности пользовательского интерфейса.   
Для программного обеспечения, которое поддерживает протокол CA/pvAccess, есть возможность считывать и записывать значения переменных процесса. Также есть пакеты расширений для обеспечения поддержки интеграции с различными языками программирования, среди которых MATLAB, Python, Tcl и т.д. Они используются в написании сценариев для взаимодействия с устройствами и объектами, управляемые с помощью EPICS.

В настоящее время данная АСНИ используется в разных проектах, среди которых: Национальный исследовательский центр синхротронного излучения (NSRRC), Китайский источник расщепляющих нейтронов (CSNS), Пекинский электронно-позитронный коллайдер (BEPC-II) и др. Основное направления в проектах связано с исследованием частиц, импульсов (или же проще говоря с экспериментальной физикой), а также с промышленностью.

Другой существующий вариант разработки в области АСНИ это TANGO (TAco Next Generation Objects). Данная система, разрабатываемая активно сообществом операторов синхротронов и воплощает в себе хорошую функциональность и технологии для работы с разнообразными объектами и компьютерным обеспечением. Важно отметить, что благодаря своим возможностям, данная система может быть использована как на всего одной, так и на нескольких машинах. С помощью сетевого протокола, основанного на реализации omniorb Zeromq или CORBA, открываются возможности различных типов связи и взаимодействия между клиентом и сервером. Это включает в себя синхронное, асинхронное (при использовании технологии CORBA) и событийно-ориентированное (Zeromq) взаимодействие.   
 Архитектура ПО TANGO основана на принципах объектно- и сервис-ориентированных подходов. Основная объектная модель TANGO включает методы, атрибуты и свойства для настройки устройств, придавая им уникальные характеристики и возможности.

Ключевыми компонентами TANGO является мощная база данных, в которой хранится вся информация о всех зарегистрированных в системе устройствах, среда для быстрой разработки приложений, а также широкий спектр вспомогательных инструментов, облегчающих эксплуатацию системы и процесс разработки.

В первую очередь весь этот инструментарий и TANGO в целом используется, чтобы обеспечить сетевой доступ к различным видам оборудования. Данный диапазон оборудования может варьироваться от простейших вариантов, таких как регистры ввода-вывода, до непростых и очень комплексных систем обнаружения или управления заводом/предприятием.

Обеспечение доступа ко всему оборудованию, осуществляется с помощью особенного процесса, называемого Сервером Устройства (Device Server). Данная концепция включает в себя устройства, принадлежащие к различным классам и обеспечивающие доступ к оборудованию. В процессе работы Сервера устройства создаются экземпляры устройств, которые представляют собой логические оборудования в системе управления. Это все обеспечивает эффективное и удобное взаимодействие клиентов с устройствами (работа с базой данных и отправка запросов).   
 На данный момент TANGO используется в таких проектах как ELI BEAMS (Чехия), ALBA (Испания), ESRF (Франция) и др. Кроме этого она используется и в России, а именно в Курчатовский Источник Синхротронного Излучения (КИСИ) и в NICA (Nuclotron-based Ion Collider Facility), это сверхпроводящий коллайдер протонов и тяжёлых ионов, город Дубна Моск. Области.

Из данных примеров можно заметить, что АСНИ используется в основном в проектах, связанных с ядерной физикой, подсчетом частиц, и импульсов, а также в промышленности. Но среди разнообразия автоматизированных систем (которые включают и АС, отвечающие за мобильную связь), их главная проблема заключается в узконаправленности, а вот АСНИ не имеет каких-то определенных границ и зачастую решает задачи АСУ, САПР и др. Но даже среди АСНИ, не было замечено систем, которые помогали бы исследовать сферу мобильной связи.

Наш проект начался именно с такой проблемы. Было предложено разработать систему для помощи с разработкой сценариев экспериментов, исследующих возможности нового стандарта мобильной связи. Наше собственное решение представляет собой автоматизированную систему научных исследований, применение которой выходит за рамки данной сферы, что означает, что её можно применять не только в области мобильной связи, но и для любых других исследований. Это возможно, так как наша система позволяет:

1. Заниматься гибко настраиваемой обработкой входных потоков данных, путём написания собственного или использования уже написанного сценария из взаимозаменяемых собственноручно закодированных или встроенных в систему стадий.
2. Трансформировать данные из входных потоков в процессе проведения эксперимента и формировать выходные потоки.
3. Формировать систему из контрольных точек для получения возможности сохранения, остановки и возобновления процессов обработки потоков данных по необходимости и желанию исследователя.
4. Сохранять параметры, индикаторы и результаты преобразований в базу данных, эффективно использовать операции обращения к данным в ней (поиск, удаление и добавление), а также делать выгрузку данных из базы в удобных для исследователя форматах.
5. Анализировать полученные и сохранённые данные, например, для сравнения эффективности разных алгоритмов, применённых на одной и той же стадии; выявление проблем и новых исследовательских вопросов, возникающих при взаимодействии определённых объектов или сценариев их поведения и другое.

В общем и целом, подводя итог вышесказанному в пунктах, система предназначена для организации (создании моделей объектов участников эксперимента, каналов связи и сценариев их взаимодействия), проведения экспериментов и анализа их результатов, а также реализации сопутствующих сервисов, в частности, для моделирования работы мобильной сети, а вообще – в любой мыслимой научной области.

* 1. Пути решения проблемы – в настоящее время это решается чисто разработкой симуляционных систем. Какие есть АСНИ и что они могут.
  2. Наш подход и наша архитектура.
     1. Графический pipeline для формирования этапов и алгоритмов обработки данных
     2. Система хранения данных, необходимых для дальнейшего анализа и сравнения результатов расчётов (преобразований)
     3. Аналитическая система для собственно сравнительного анализа результатов различных экспериментов
  3. Описание платформы разработки и используемых систем.

1. Подзадача: её описание, роль в общей системе и общая идея реализации.
   1. Характеристика
   2. Требования (функциональные и нефункциональные)
   3. Идея и подробности имплементации
   4. Небольшие примеры (если это имеет смысл)
2. Реальный пример использования всей системы
3. Заключение
   1. Итоги разработки
   2. Направления дальнейшего развития и дальнейших разработок

**Экономическое обоснование**

Важнейшей особенностью нашего проекта является его открытый исходный код, который было решено распространять под одной из свободных лицензий. В следствии того, что весь процесс сборки описывается в проектной документации, можно сказать, что исполняемые файлы и библиотеки, включенные в состав нашего проекта, также распространяются по открытой лицензии.

С формальной точки зрения, свободная лицензия – это лицензионный договор, по которому правообладатель безвозмездно предоставляет пользователям разрешение на использование произведения любыми способами, в том числе путем распространения и создания производных произведения.

Лицензии делятся на 2 типа – копилефтные и разрешительные. Копилефтной является лицензия, требующая распространять производные продукты под такой же лицензией. Разрешительная лицензия разрешает любое возможное использование продукта.

Ниже приведу список наиболее широко используемых открытых лицензий с их кратким описанием:

1. «GPLv3» - копилефтная лицензия, то есть требует, чтобы исходные коды производных работ были открытыми под ней же. Другими словами, если в проекте используется сторонняя библиотека, имеющая лицензию «GPLv3», тогда проект должен наследовать ее же;
2. «LGPLv3» - надстройка над «GPLv3», и требует наличия текстов обеих лицензий в проекте. Отличается от «GPLv3» тем, что библиотеки под этой лицензий разрешается использовать для создания программ под другими лицензиями путем компоновки;
3. «MIT License» - разрешительная лицензия. Разрешает использование кода практически любым образом, кроме того, что текст лицензии и указание авторства не будут стерты;
4. «BSD License» различных версий. Является разрешительной лицензией, по смыслу аналогичной с «MIT License»

Важнейшим нюансом в выборе открытой лицензии для своего программного обеспечения является лицензия стороннего ПО, которое было использовано в проекте. Библиотеки, которые мы использовали в своем проекте, имеют следующие лицензии:

1. «Qt 5.12.12» - для проектов с открытым исходным кодом может распространяться одновременно по двум копилфетным лицензиям: «GPLv3» и «LGPLv3», одну из которых требуется унаследовать. Выбор лицензии во многом зависит от используемых компонентов. Например, компонент библиотеки «Qt Charts» не распространяется по «GPLv3», и если проект будет использовать этот компонент, тогда он должен наследовать лицензию «GPLv3»;
2. «libpq» - распространяется по проприетарной лицензии, очень похожей на «MIT License» и «BSD License», соответственно, производный проект может ее не наследовать;
3. «Google Test» - распространяется по «BSD 3-Clause»;
4. «yaml-cpp» - распространяется по «MIT License».

В нашем проекте была использована библиотека «Qt Charts», лицензируемая под «GPLv3». В силу того, что лицензия является копилефтной, мы унаследовали ее в нашем проекте.

**Наш подход и наша архитектура**

Нашей задачей в рамках данной работы является разработка АСНИ для моделирования и анализа сценариев работы мобильной сети. Система предназначена для организации и проведения экспериментов по работе с мобильной сетью, а также для анализа и сопоставления результатов этих экспериментов. Принципиально система состоит из двух модулей:

* Интегрированная среда разработки
* Интегрированная среда моделирования

В свою очередь, интегрированная среда моделирования состоит из следующих подмодулей:

* Пользовательский интерфейс для взаимодействия со средой моделирования
* Подсистема управления расчетом эксперимента
* Подсистема анализа результатов проведения эксперимента
* Подсистема хранения данных

Для реализации пользовательского интерфейса интегрированной среды моделирования было решено воспользоваться парадигмой визуального программирования – это такой способ написания программ, при котором программисту не требуется писать текст программы, а требуется производить манипуляции с графическими блоками, каждый из которых выполняет какое-то действие. Таким образом, исследователь будет разрабатывать программу эксперимента, манипулируя графическими блоками вместо написания её текста, что должно упростить исследование эксперимента.

С точки зрения подсистемы управления расчетом эксперимента каждый графический блок в интегрированной среде моделирования эксперимента представляет собой специальный объект “стадии” работы с данными. Каждая стадия является отдельной подпрограммой, которая может принимать какие-либо входные параметры, производить с ними произвольные манипуляции, после чего возвращать выходные данные. Стадии могут выстраиваться в ориентированные графы таким образом, что выходные данные одной или нескольких стадий могут служить входными данными для одной или нескольких стадий. Это позволяет исследователю реализовывать сценарии произвольной сложности, не задумываясь при этом о деталях реализации отдельно взятых стадий, а также, не заботясь о распараллеливании входных и выходных потоков данных при их обработке.

Помимо того, что стадии можно собрать в программу и запустить её, стадии поддерживают и интерактивную работу, полезную для отладки и анализа: исследователю доступна возможность приостановить выполнение любой из стадий и проанализировать входные и выходные потоки данных этой стадии, после чего исследователь может по своему усмотрению прервать выполнение программы или продолжить её с того места, где она остановила своё выполнение.

Подсистема анализа результатов проведения эксперимента позволяет исследователю анализировать конечные и промежуточные результаты проведения эксперимента. Для этого исследователю нужно добавить в эксперимент стадии для анализа данных. Эти стадии идеологически ничем не отличаются от стадий расчетов эксперимента – они также принимают некоторые входные данные, обрабатывают их и возвращают выходные данные, представляющие собой требуемый результат анализа входных данных. Таким образом, исследователю доступна возможность параллельно проводить эксперимент и при этом анализировать некоторые промежуточные результаты этого эксперимента, подключая стадии анализа данных к промежуточным стадиям эксперимента.  
Помимо этого подсистема анализа результатов проведения эксперимента позволяет анализировать и сопоставлять промежуточные и конечные результаты проведенных ранее экспериментов.

Подсистема хранения данных позволяет исследователю сохранять конечные и промежуточные результаты проведения экспериментов. Для этого исследователю нужно добавить в эксперимент стадию для сохранения данных. Эта стадия позволяет сохранить произвольные данные для их последующего использования. Например, исследователь, сохранив выходные данные проведения одного эксперимента, может их использовать в качестве входных данных для проведения другого эксперимента. Также, сохранив результаты одного или нескольких экспериментов, исследователь в дальнейшем может их проанализировать, используя подсистему анализа результатов проведения экспериментов. Таким образом, подсистема хранения данных может обеспечивать дополнительное взаимодействие подсистемы управления расчетом эксперимента и подсистемы анализа результатов проведения эксперимента.

На этом закончим рассмотрение интегрированной среды моделирования и перейдем к рассмотрению интегрированной среды разработки. Предполагается, что интегрированной средой моделирования пользуется инженер по моделированию – то есть исследователь, который исследует некоторый эксперимент, в то время как интегрированной средой разработки пользуется инженер по разработке для разработки новых стадий обработки данных. В настоящий момент в качестве интегрированной среды разработки выбрана платформа Microsoft Visual Studio Express 2017. Инженер по разработке, имея полный код нашего проекта и подсистемы управления расчетом эксперимента в частности, может добавить в последнюю новые стадии обработки данных, необходимых исследователю.

**Описание платформы разработки и используемых систем**

Компиляторы и средства сборки:

В качестве компилятора был выбран “*Microsoft Visual C++*”, поставляемый вместе с бесплатной интегрированной средой разработки “*Mircosoft Visual Studio Express 2017*”. Данная версия компилятора и поставляемая с ним стандартная библиотека шаблонов (STL (Standard Template Library)) поддерживает стандарт C++17, который было решено использовать, как самый новый из тех, что могут быть доступны при использовании.

Помимо интегрированной среды разработки “*Microsoft Visual Studio*”, для прототипирования GUI была также использована IDE “*Qt Creator*”, ключевая особенность которого состоит в наличии редакторе форм, позволяющим добавлять в приложение элементы графического интерфейса без редактирования кода.

Для сборки проекта было принято решение использовать “*CMake*” – кроссплатформенное программное средство для автоматической сборки программы из исходного кода. Решение принято, исходя из распространенности “*CMake*” в современных проектах на языках C и C++, а также из намерения введения будущей поддержки кроссплатформенности для запуска АСНИ на различных операционных системах и аппаратных платформах.

Система управления базами данных (СУБД):

Для хранения данных, генерирующихся в ходе проведения эксперимента, было решено использовать “*PostgreSQL*” – кроссплатформенная объектно-реляционная система управления базами данных с открытым исходным кодом.

Используемые библиотеки:

1. “*Qt 5.12.12*” – набор библиотек для разработки кроссплатформенного программного обеспечения на языке программирования C++. Из поставляемого набора были использованы библиотеки “*Qt5 Core*”, ”Qt*5 GUI*”, “*Qt5 Charts*” , “*Qt5 Widgets*”.

|  |  |
| --- | --- |
| Название библиотеки | Содержание библиотеки |
| Qt5 Core | Набор классов, напрямую не связанных с GUI и используемых во всех остальных библиотеках Qt. Содержит метаобъекты для слотов и сигналов, контейнеры, например, QVector, QLine, QPoint, QMap, QRect. В библиотеку также включена событийная система QEvent. |
| Qt5 GUI | Низкоуровневые базовые классы, используемые для разработки компонентов графического интерфейса. В рамках проекта библиотека используется виджетами для рисования: QPixmap, QPainter. |
| Qt5 Charts | Компоненты графического интерфейса, используемые для построения и отображения графиков на основе статических или изменяющихся в процессе исполнения данных. Используется для отрисовки графиков в дашборде. |
| Qt5 Widgets | Содержит все компоненты оконного приложения, а также QApplication, который представляет из себя само оконное приложение, фактически, являющееся виджетом. |

1. “Google Test” – самый распространенный фреймворк с открытым исходным кодом для модульного тестирования на языке C++. Предоставляет следующие возможности:
   1. Создание одиночных тестов (тест-кейсов). Тест-кейс определяется синтаксисом “TEST(GroupName, TestName)”, где “*GroupName*” – название тестовой группы, “ *TestName* ” – наименование тест-кейса;
   2. Группировка тестов – тесты могут быть объединены в единые наборы по выбранному программистом критерия. Самым распространенным способом объединения является группировка тестов по классу, над которым проводится тестирование;
   3. Создание тестовых классов. Тестовый класс представляет из себя единую конфигурацию объектов, которая может быть повторно использована в тестах или наборах тестов. Тест-кейс, использующий тестовый класс должен определяться через макрос “TEST\_F(TestFixtureClassName, TestName)”, где “*TestFixtureClassName*” – имя того класса, который подразумевается использовать, а “TestName” – имя тест-кейса;
   4. Набор макросов, используемых для проверки утверждений. Проверки делятся на 2 типа: “Assertions” и “Expectations” и выражаются в коде через макросы, представляющие из себя регулярное выражение “ASSERT\_\*” или “EXPECT\_\*” соответственно. Одним из множества примеров таких макросов является “ASSERT\_EQ(val1, val2)” и “EXPECT\_EQ(val1, val2)”. Он проверяет, что val1 и val2 равны между собой. Различием между “ASSERT\_\*” и “EXPECT\_\*” является поведение тестовой функции после исполнения кода макроса. В случае неисполнения утверждения в “ASSERT\_\*”, тестовая функция завершит свое исполнение, а тест-кейс будет помечен, как проваленный. Если ожидание в функции “EXPECT\_\*” будет провалено, тогда тестовая функция продолжит исполнение, но при окончании тест также будет завершен с ошибкой;
   5. Создание mock-классов и имитация поведения интерфейсных функций. Mock-классы используются для имитирования поведения составных частей тестируемого класса. Они необходимы для тестирования класса при различных сценариях поведения его составных частей, связанных с ним отношениями композиции и агрегации.   
      Mock-класс определяется, как наследник базового класса, в котором все интерфейсные методы, поведение которых должно быть имитировано, заносятся под макрос “MOCK\_METHOD”.
2. “*libpq*” – библиотека с открытым исходным кодом для взаимодействия с СУБД “*PostgreSQL*”. Предоставляет набор функций, написанных на языке C и используемых для формирования запросов и транзакций к СУБД. Важно отметить, что существует еще две распространенные библиотеки для работы с СУБД “*PostreSQL*”: “*libpq++*” и “*libpqxx*”, которые предоставляют интерфейс на языке C++. “libpq” был выбран, так как в сравнении с библиотеками на C++ имеет полную и подробную документацию.
3. “yaml-cpp” – библиотека с открытым исходным кодом для чтения и записи данных в YAML-формате. YAML (Yet Another Markup Language, еще один язык разметки) – это специальный язык структурированной записи информации, имеющий текстовый формат и используемый для сериализации объектов.   
   Все узлы YAML-файла представляются в библиотеке, как экземпляры класса «YAML::Node». Для загрузки корневого узла документа необходимо вызвать функцию «YAML::Node YAML::LoadFile(const char\* filePath)», где «filePath» – путь до YAML-документа. В библиотеке определены итераторы для обхода дочерних узлов итерируемой ноды. Объект, находящейся в итераторе, содержит два поля: first и second. В поле «first» хранится наименование текущего узла, в поле «second» – значение узла. Для использования значений узлов, их необходимо конвертировать с использованием метода «template <typename T> YAML::Node::as<T>()». Если текущий узел невозможно привести к типу «T» - библиотека выбрасывает исключение.  
   Для записи в файл библиотека предоставляет класс «YAML::Emitter». Эмиттер накапливает и форматирует данные, которые в него записываются, и предоставляет метод для получения конечной форматированной в YAML-формате строки: «YAML::Emitter::c\_str()».