Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Контрольная работа №1 по ТРПО

«Школьный журнал»

Выполнил: Бакир Манар Манхалович

студент 2 курса, специальность ПОИТ

группа № 581072

Домашний адрес: ул. Ландера, кв. 87

Телефон: +375291087332

Электронный адрес: bakirmanar@mail.ru

Проверила: Бакунова Оксана Михайловна

Минск, 2017

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4

1.1 Постановка задачи 4

1.2 Анализ требований к автоматизированной системе 5

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА. 6

3 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ 9

3.1 Диаграмма вариантов использования 9

3.2 Диаграмма классов 10

3.3 Диаграмма деятельности 11

3.4 Диаграмма коопераций 11

3.5 Диаграмма последовательности 12

3.6 Диаграмма развёртывания 13

3.7 Диаграмма компонентов 14

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16

ГЕНЕРАЦИЯ КОДА 17

**ВВЕДЕНИЕ**

В настоящее время во все сферы деятельности человека широко внедряются информационные технологии. Это приводит к разработке огромного количества программных средств (ПС) различного функционального назначения. При этом объем и сложность используемых ПС постоянно возрастают. Возрастает значительность информации в самом широком смысле этого слова и связанных с ней информационных технологий. Для больших объемов информации главным качеством является их структурированность, так как именно от характера структуры данных зависит скорость обработки поиска информации.

База данных на сегодняшний день – это самый распространенный «повод» для написания программ. Все современные языки программирования содержат в себе встроенные возможности для быстрого и удобного создания СУБД.

СУБД – это, конечно, нечто большее, чем просто набор структурированных данных. СУБД можно назвать «умной» средой, с помощью которой можно обрабатывать, искать, передавать, хранить данные. От качества СУБД зависит эффективность работы с базой данных в целом. Современные СУБД – это «высокоинтеллектуальные» системы, позволяющие работать не только с базами данных, но и с базами знаний. Это направление, связанное с накоплением, получением, сортировкой и использованием знаний, является довольно новым. С понятием «база знаний» тесно связанно понятие искусственного интеллекта, что говорит об огромных масштабах работ, которые потребуются провести для совершенствования сегодняшних СУБД, так как искусственный интеллект – это новый горизонт в нашей компьютерной эпохе. При работе баз данных и связанных с ними программ обработки и поиска информации необходимо учитывать специфику предметной области – одно из важнейших достоинств базы, данных. Конечно, здесь можно говорить о том, что уже давно существуют универсальные формы для баз данных, универсальные СУБД, универсальный модуль обработки данных. Но эта универсальность требует больших, (если не огромных) затрат машинного времени и ресурсов, не говоря уж о стоимости всего ПО. Поэтому чаще всего используют специализированные БД и СУБД.

Реляционная модель СУБД основывается на математическом понятии отношения, а общая структура данных представляется в виде таблицы, в которой каждая строка значений соответствует логической записи, а заголовки столбцов являются характеристиками объектов, информацию о которых необходимо хранить в БД.

Процесс проектирования БД представляет собой последовательность перехода от неформального словесного описания информационной структуры предметной области к формальному описанию объектов предметной области в терминах некоторой модели.

В данной курсовой работе также будет создана специализированная база данных для учета успеваемости и посещаемости учащихся в школе.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Постановка задачи

Люди с ранних лет обучаются в школах. Целью обучения в школах является получение общего образования. Также школы бывают с различными уклонами, например: с музыкальным уклоном, с информатическим уклоном и др.

Обучение длится до 11 лет. После окончания 4 лет обучения, человек получает начальное образование. После окончания 9 лет обучения – базовое образование. После окончания 11 лет обучения – среднее образование.

Каждый учебный год делится на 4 периода обучения – четверти.

Всех учащихся на разделяют на отдельные группы в зависимости от их количества. Эти группы называют классами. Некоторые классы могут иметь различные уклоны, например: класс с музыкальным уклоном, класс с правовым уклоном и др.

Учащиеся посещают занятия(уроки), которые длятся 45 минут. Они проводятся в классных помещениях, а также, при необходимости, в спортзале или на улице. Каждое занятие посвящено какому-либо предмету. Ведет занятие квалифицированный работник – учитель.

На занятии учитель рассказывает новую тему, показывает различные примеры. Часто вызывает нескольких учащихся, для того, чтобы ответить вопрос по теме, рассказать тему или решить задачу. Изредка учащиеся пишут проверочные или контрольные работы.

Работа учащегося на занятии может оцениваться в зависимости от его ответов. Все учащиеся получают оценки за проверочные и контрольные работы.

В конце четверти учащиеся получают по каждому предмету четвертную оценку, являющуюся средним баллом от всех полученных за эту четверть оценок.

В конце учебного года учащиеся получают по каждому предмету годовую оценку, являющуюся средним баллом от всех четвертных оценок.

Учитель ведет учет всех оценок и посещаемости в классный журнал.

Анализ предметной области позволил выявить минимальный набор из девяти сущностей:

1. Школа – [учебное заведение](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%87%D1%80%D0%B5%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5) для получения [общего образования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D0%B5_%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).
2. Ученик – человек, получающий образование в школе.
3. Учитель – квалифицированный работник школы, обучающий учеников.
4. Занятие (урок) – период времени в 45 минут, посвященный обучению какому-либо предмету.
5. Предмет – часть [объекта](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), определённый его аспект, исследуемый в каком-либо конкретном случае.
6. Оценка – выраженное в числе мнение преподавателя (другого проверяющего лица) об уровне знаний ученика (качестве его работы).
7. Учебный год – период длительностью в один обычный год, обозначающий годичный цикл учебного процесса в образовательных учреждениях
8. Четверть – один из 4 учебных периодов, из которых состоит учебный год.
9. Журнал - основной государственный документ регистрации посещаемости и записи отметок в школах.
   1. Описание и обоснование выбранного языка программирования

C++ — компилируемый статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Процесс проектирования архитектуры программного обес­печения состоит в проектировании структуры всех его ком­понент, функционально связанных с решаемой задачей, включая сопряжения между ними и требования к ним.

Архитектура программного обеспечения в традиционном смысле включает определение всех модулей программ, их иерархии и сопряжения между ними и данными.

Во время разработки архитектуры программного обеспечения выполняется его модульно-иерархическое построение.

1. **СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило — наиболее важная функция находится в верхнем левом углу, кроме того есть правило стороны:

* стрелка входа приходит всегда в левую кромку активности,
* стрелка управления — в верхнюю кромку,
* стрелка механизма — нижняя кромка,
* стрелка выхода — правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-проектов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия:

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде прямоугольника (см. рис. 2.1) и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, “производить услуги”, а не “производство услуг”).

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой мы называем универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друх от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира. Примером сущности IDEF1X может быть сущность "СОТРУДНИК", которая представляет собой всех сотрудников предприятия, а один из них, скажем, Иванов Петр Сергеевич, является конкретной реализацией этой сущности.

Контекстная диаграмма модели показана на рисунке 2.1. Декомпозиция контекстной диаграммы функциональной модели показана на рисунке 2.2. Для построения модели использовался продукт BPwin 4.0 фирмы Computer Associates.

Субъектом моделирования является программное средство «Школьный журнал», цель – описать функциональность работы программного средства, точка зрения – преподаватель, учащийся.



Рисунок 2.1 – Контекстная диаграмма модели

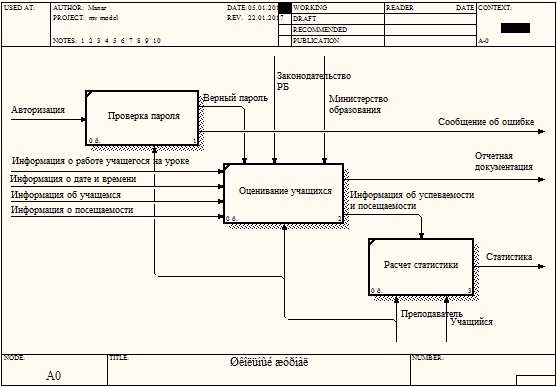


Рисунок 2.2– Декомпозиция контекстной диаграммы функциональной модели

# ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ

## Диаграмма вариантов использования

Модель вариантов использования предназначается для определения требований к системе. Она включает в себя актеров, варианты использования и связи между ними. Для отображения этой модели язык UML предлагает использовать диаграммы Use Case (вариант использования) совместно с моделями State Diagram (диаграммы состояний) и Activity Diagram (диаграммы деятельности/активности). Последние используются для конкретизации вариантов использования системы

Разработка диаграммы вариантов использования преследует цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования представлена на рисунке 3.1.

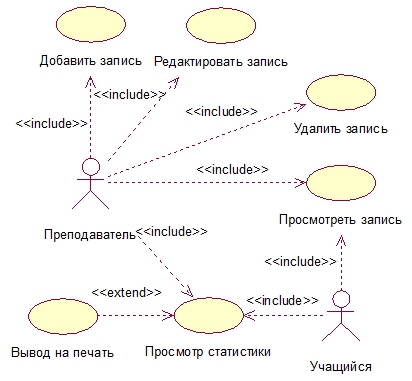


Рисунок 3.1 – Диаграмма вариантов использования

## Диаграмма классов

Диаграмма классов — диаграмма, демонстрирующая классы системы, их атрибуты, методы и взаимосвязи между ними. Входит в UML.

Существует два вида:

* статический вид диаграммы рассматривает логические взаимосвязи классов между собой;
* аналитический вид диаграммы рассматривает общий вид и взаимосвязи классов, входящих в систему.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

* концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
* точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
* точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма классов представлена на рисунке 3.2.

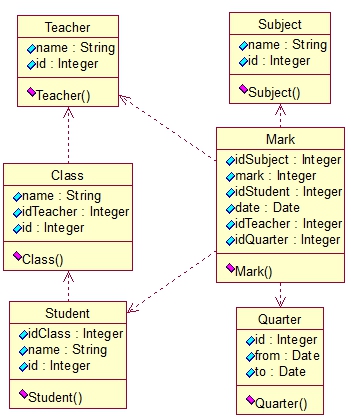


Рисунок 3.2 – Диаграмма классов

## Диаграмма деятельности:

Диаграмма деятельности — UML-диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграмма деятельности представлена на рисунке 3.3.

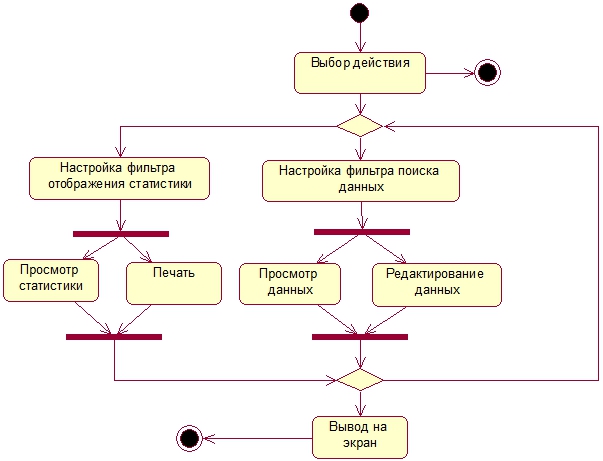


Рисунок 3.3 – Диаграмма деятельности

## Диаграмма коопераций:

Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме кооперации в виде прямоугольников изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи – потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. На этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров. Следовательно, если необходимо явно специфицировать взаимосвязи между объектами в реальном времени, лучше это делать на диаграмме последовательности.

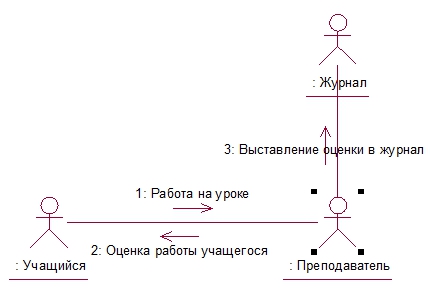


Рисунок 3.4 – Диаграмма коопераций

## Диаграмма последовательности:

Диаграмма последовательности – диаграмма, на которой показано взаимодействие объектов, упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления. Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Диаграмма последовательности представлена на рисунке 3.5.

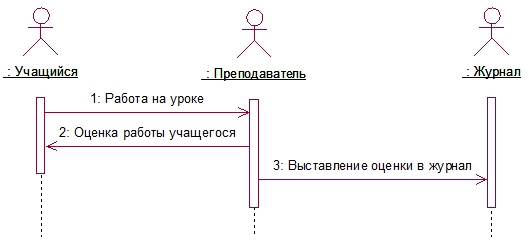


Рисунок 3.5 – Диаграмма последовательности

## Диаграмма развёртывания:

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь под узлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

* Узел устройства;
* Узел среды выполнения.

Узлы устройств – это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения – это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

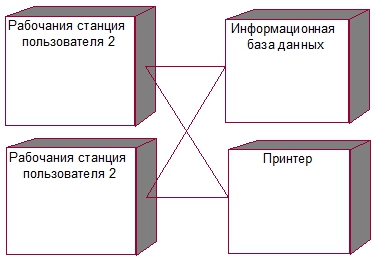


Рисунок 3.6 – Диаграмма развёртывания

## Диаграмма компонентов:

Диаграмма компонентов – статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемые и требуемые интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывается связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

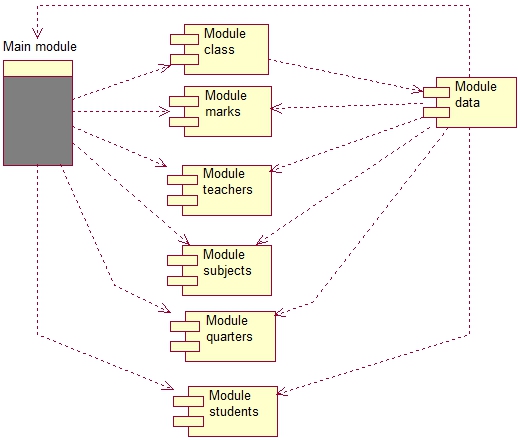


Рисунок 3.7 – Диаграмма компонентов

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. ГОСТ 2.105-95 ЕСКД. Введ. 1996–07–01. – М.: Изд-во стандартов, 1996.
2. ГОСТ 19.104–78 – Единая система программной документации. Основные надписи.
3. Маклаков С.В. BPwin и ERwin: CASE - средства для разработки информационных систем.
4. Федотов Д.Э., Семенов Ю.Д., Чижик К.Н. Практикум для высших учебных заведений. CASE-технологии.-157с.
5. Интернет обучение [Электронный ресурс] / Котляр Д.С. Использование CASE-средства ERwin для автоматизации проектирования и разработки базы данных – Режим доступа: http://royallib.com/read/ bezopasnost – Дата доступа: 15.12.2015.
6. Интернет обучение [Электронный ресурс] / Информационные системы и технологии – Режим доступа: <http://www.narfu.ru> – Дата доступа: 19.12.2015.
7. Интернет обучение [Электронный ресурс] / Нормализация структурны данных – Режим доступа: http://infostart.ru/public/269803/ – Дата доступа: 15.01.2016.
8. Интернет обучение [Электронный ресурс] / Программа компьютерного моделирования BpWin – Режим доступа: http://bourabai.kz/cm/bpwin.htm– Дата доступа: 11.11.2015.

# ГЕНЕРАЦИЯ КОДА

На рисунках 4.1 – 4.3 представлен процесс генерации кода на языке C++, непосредственно сам код расположен ниже.

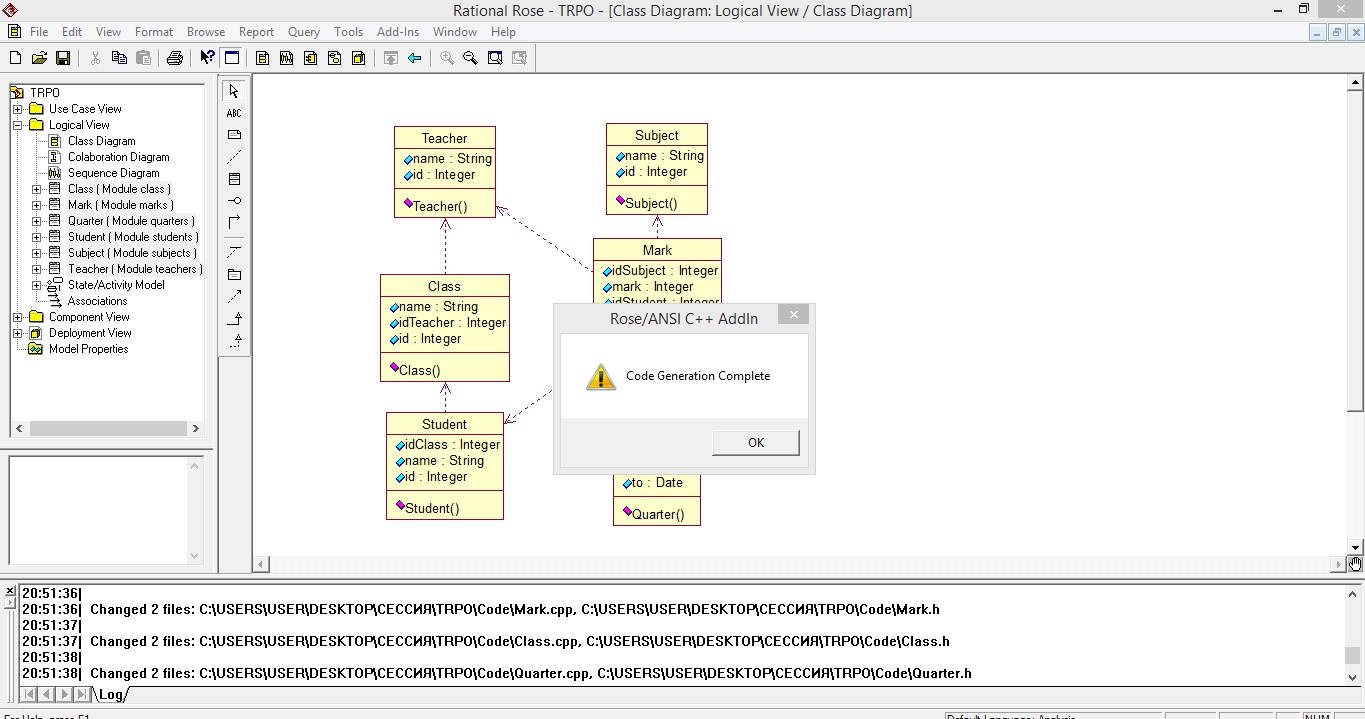


Рисунок 4.1 – Процесс генерации кода

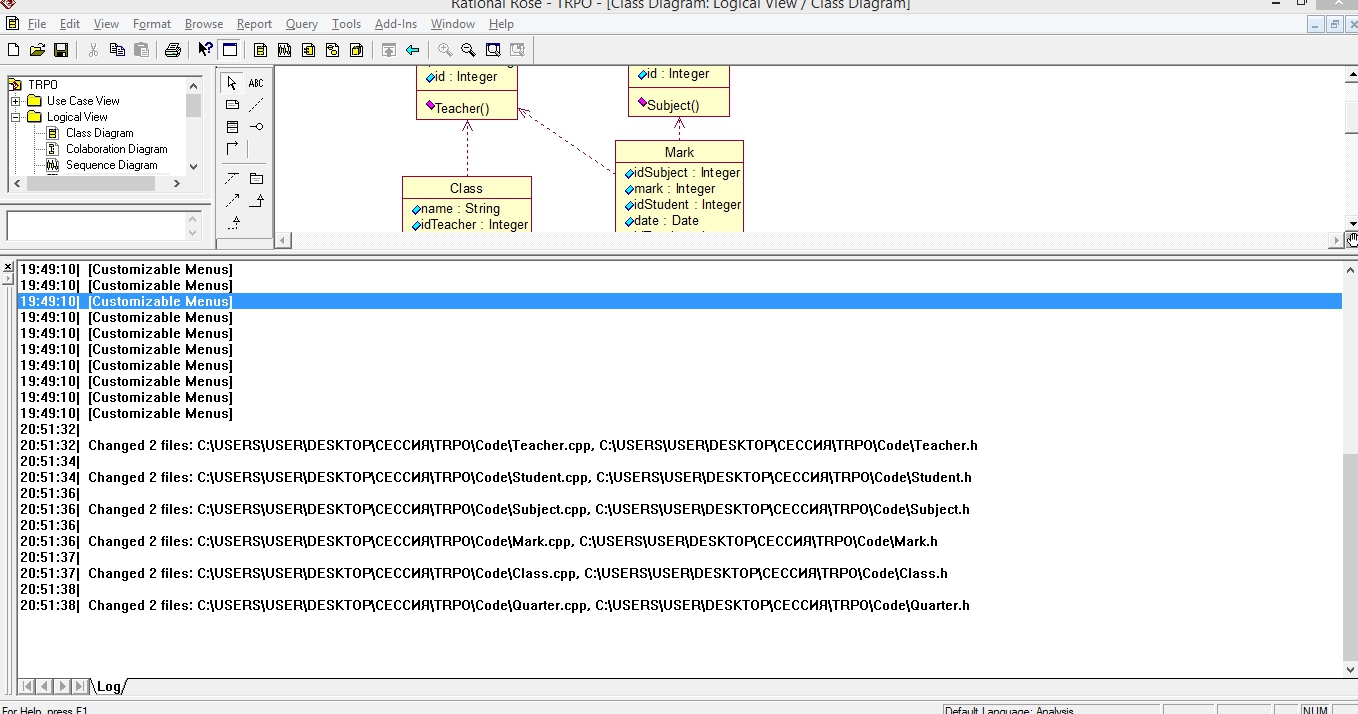


Рисунок 4.2 – Процесс генерации кода

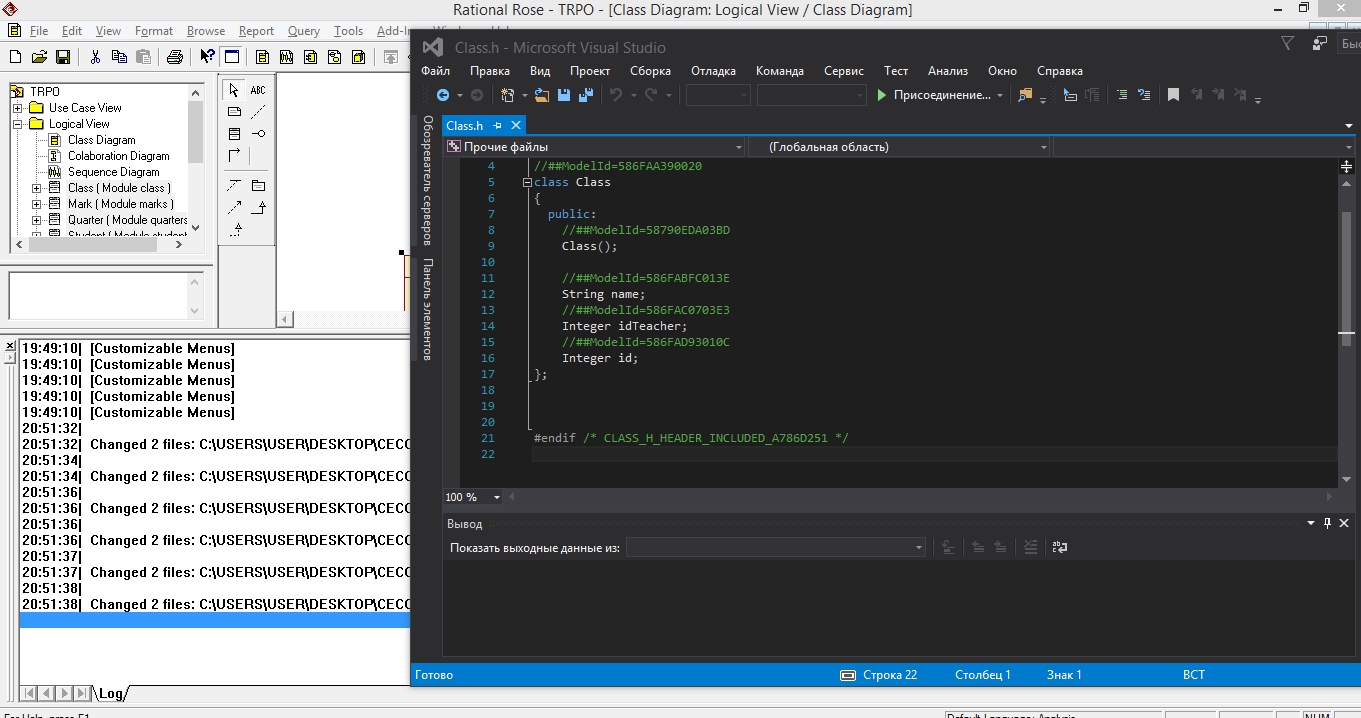


Рисунок 4.3 – Процесс генерации кода

Сгенерированный код:

class Subject

{

public:

Subject();

String name;

Integer id;

};

class Student

{

public:

Student();

Integer idClass;

String name;

Integer id;

};

class Quarter

{

public:

Quarter();

Integer id;

Date from;

Date to;

};

class Mark

{

public:

Mark();

Integer idSubject;

Integer mark;

Integer idStudent;

Date date;

Integer idTeacher;

Integer idQuarter;

};

class Class

{

public:

Class();

String name;

Integer idTeacher;

Integer id;

};

class Teacher

{

public:

Teacher();

String name;

Integer id;

};