**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Федеральное агентство по образованию**

**Государственное образовательное учреждение**

**высшего профессионального образования**

**“Удмуртский государственный университет”**

**Факультет “Информационных технологий и вычислительной техники”**

**Выпускная Квалификационная работа**

**Тема: «Разработка приложения – оболочки для генерации задач»**

Выполнил:

студент группы 38-51 ФИТиВТ.

специальность «Информационные системы»

Кононов Антон Александрович

Научный руководитель:

старший преподаватель кафедры Математики и Информатики

Трусов Андрей Сергеевич

Ижевск 2008 г.

Содержание

[1. Введение. 2](#_Toc200135938)

[2. Словарь терминов и сокращений 4](#_Toc200135939)

[3. Постановка задачи. 5](#_Toc200135940)

[4. Теоретическая часть 6](#_Toc200135941)

[4.1. Проблема постановки задачи для компьютера. 6](#_Toc200135942)

[4.2. Обзор существующих решений по генерации задач. 9](#_Toc200135943)

[4.3. Распространение и развитие разработки. 11](#_Toc200135944)

[4.4. Алгоритм, позволяющий параметризовать задачу. 12](#_Toc200135945)

[4.5. Реализация дополнительных алгоритмов генерации. 17](#_Toc200135946)

[4.6. Программные средства. 18](#_Toc200135947)

[4.7. Программная платформа .NET особенности, архитектура. 19](#_Toc200135948)

[4.8. Преимущества и недостатки новой платформы в разработке. 25](#_Toc200135949)

[5. Практическая часть. 31](#_Toc200135950)

[5.1. Для кого эта программа. 31](#_Toc200135951)

[5.2. План разработки. Выбор средств. 33](#_Toc200135952)

[5.3. Вопросы расширяемости и распространения системы. 38](#_Toc200135953)

[5.4. Описание системы. 40](#_Toc200135954)

[6. Заключение 49](#_Toc200135955)

[6.1. Отчет по решенным и нерешенным задачам. 49](#_Toc200135956)

[6.2. Взгляд в будущее. 50](#_Toc200135957)

[7. Список литературы 51](#_Toc200135958)

[8. Приложение 1. Описание интерфейсов программы 52](#_Toc200135959)

[9. Приложение 2. Лицензия GNU GPL 56](#_Toc200135960)

# Введение.

За последние 10 лет сфера образования сделала огромный рывок навстречу информационной эпохе. Казалось бы, совсем недавно персональный компьютер был роскошью в доме, а в образовательных учреждениях компьютеры были только у директоров и руководящего состава. Наличие компьютерного класса говорило о высоком статусе заведения и многие родители только благодаря этому решались отдать своих детей в такое место на воспитание. С того времени изменилось очень многое. Персональные компьютеры стали доступны не только семьям с хорошим заработком, но и среднему классу. За последние два года компьютерными классами были оснащены множество средних школ и лицеев. Все вновь отстроенные школы также имеют хорошо обустроенные компьютерные классы. Еще сильнее свое влияние информационная сфера оказала на высшие учебные заведения, где все, от абитуриента до профессора пользуются персональными компьютерами для работы и учебы.

Естественно, что компьютер становится неотъемлемым помощником в образовании. Глобальная сеть Интернет предоставляет доступ к огромному количеству информации, ведь именно усвоение новой информации и получение знаний есть основная цель обучения для студентов. Здесь естественным образом есть место и для преподавателей, в Интернете существует огромное количество научных статей и работ, научных сообществ. Теперь уже не нужно публиковать работы в дорогостоящих журналах или газетах, достаточно разместить их в электронном варианте в сети Интернет.

Постепенно образование переходит на новую ступень. Вводятся новые дисциплины, новые методики преподавания и новые стандарты. Так, совсем недавно был введен «Федеральный Интернет-экзамен в сфере профессионального образования» (ФЭПО), который представляет собой тестирование по профильному предмету, с целью выявления общей подготовки студента. Во многих ВУЗах оценка по такому тестированию уже принимается в качестве оценки по соответствующему профильному предмету. Нет никакого сомнения, что весь этот процесс направлен на постепенный переход современного российского образования к западной методике тестирований. Быть может кто-то действительно против таких преобразований, но бороться с ними уже не имеет смысла, правильнее будет влиться в новое русло и внести свой вклад в развитие образования в информационной сфере.

Для преподавателя нет ничего важнее контроля знаний и успеваемости его подопечных. Ведь именно передача знаний, диалог между преподавателем и студентом является целью обучения для преподавателя. Конечно, во многих гуманитарных науках такой проверкой становится демонстрация студентом своих знаний при помощи размышлений на различные темы. Однако уже сейчас все большее предпочтение отдается тестированию студентов на наличие точных знаний. Естественно составление таких тестов требует от преподавателя огромных усилий, ведь помимо составления вопросов он сам должен знать ответы на каждый из них и быть уверенным, что в самих вопросах нет двусмысленности. Для многих преподавателей составление тестовых заданий давно превратилось в рутинное занятие. Особенно эта проблема ощутима у преподавателей естественных наук: математики, физики, алгебры и т.д. В их заданиях нередко разнообразие заданий заключается лишь в разнообразии вариантов, в то время как сами задания представляют собой полные или частично измененные копии друг друга. Конечно, такие задания позволят проверить знания всех студентов на одном уровне, с другой стороны при решении таких заданий более слабые студенты чаще проявляют свою наиболее развитую способность к списыванию. Несомненно, создание практических заданий определенной сложности и разнообразия влечет за собой более кропотливую работу преподавателя, отнимает немало времени и сил. Если учесть, что на одного преподавателя в день приходится по 3-4 группы студентов, то подготовка ко всем занятиям может занять больше времени, чем сами занятия. В то же время многие преподаватели уже разработали некоторую методику по созданию тестовых заданий. Одни выбирают из уже существующих заданий наиболее подходящие, и модифицируют их при необходимости. Другие наоборот придумывают новые задания на основе общей тематики теста и набора алгоритмов (формул). Наиболее «продвинутые», то есть обладающие определенными навыками и знаниями, в области информационных технологий преподаватели уже создали, либо пытаются создать алгоритм для компьютерной программы, которая будет создавать задания определенной тематики. Но и таким преподавателям нужна помощь в организации и систематизации своих собственных наработок.

Таким образом, мы видим, что перед нами стоит несколько взаимосвязанных задач, решение которых позволит сэкономить время преподавателя, помочь ему в организации накопленных решений и тем самым повысить эффективность всего процесса образования, направив свободное время преподавателя на диалог с учеником, а не на рутинную работу.

# Словарь терминов и сокращений

* ПО – программное обеспечение.
* ПК – персональный компьютер
* ФЭПО – Федеральный интернет-экзамен в сфере профессионального образования.
* MS - Microsoft. ®
* СУБД – система управления базами данных
* CLR – common language runtime
* MSIL – Microsoft intermediate language.
* PE – portable executable
* XML – extensible markup language

# Постановка задачи.

Первоначально целью данной работы ставилась разработка ПО для хранения и систематизации методического материала по математическим дисциплинам студентов 1-2 курсов. В процессе исследования проблемы, мной была выявлена не менее важная проблема, касающаяся в большей степени самих преподавателей. Таким образом, было решено разработать приложение, которое поможет не только студентам, но и преподавателям.

Целью этой работы стала разработка компьютерного приложения, которое позволит преподавателям в полуавтоматическом режиме генерировать задания для проверки знаний по различным дисциплинам. Также данное приложение должно расширяться за счет алгоритмов, написанных самими преподавателями. Все задачи и шаблоны задач должны систематизироваться за счет разделов и подразделов. В приложении должна быть возможность сохранения методических материалов по отдельным задачам или разделам. Особенно важно реализовать вывод текстов готовых задач в файлы для печати, реализовав при этом несколько форматов (просто текст, MS Word®, TEX). Весь функционал должен складываться в единое приложение, с интуитивно понятным интерфейсом и стабильным в работе. Хранение информации было решено реализовать в обычном файле, чтобы облегчить процесс переноса приложения или просто информации. Само приложение должно служить определенного рода помощником для преподавателя в составлении и систематизации алгоритмов и типов заданий.

Решение всего проекта было принято реализовать в три этапа:

1. Теоретический анализ проблемы.
2. Выбор инструментов разработки, удовлетворяющих поставленной задаче.
3. Разработка приложения.

Хочется отметить, что целью работы не является разработка некоего универсального генератора задач, как могло показаться. Основной целью ставится создание своего рода оболочки для хранения накопленного материала, одной из возможностей которой будет генерация заданий на основе простых алгоритмов.

Целевой аудиторией по применению данного приложения могут стать как преподаватели школ, так и преподаватели начальных курсов высших учебных заведений, т.к. именно в это время практическая нагрузка по решению тестовых и небольших заданий наиболее высока у обучающегося, а, следовательно, на это уходит довольно много времени и самого преподавателя. Применение подобной системы поможет преподавателям не только собрать все накопленные знания в единое целое, но и сохранить собственное время, переложив часть работы на плечи компьютера. Также эта система поможет решить проблему списывания решений между учениками одной и даже разных параллелей.

# Теоретическая часть

## Проблема постановки задачи для компьютера.

Как уже было отмечено, в данной работе мы не стремимся создать универсальный генератор задач. Создание подобного генератора – задача трудоемкая и требующая определенных материальных вложений. Вообще говорить сегодня о некоем универсальном генераторе рано, поскольку компьютер может работать только с числами, то мы не сможем сгенерировать задачу, допустим, по философии или литературе. Однако, генерация задач во многих современных программах, в том числе и федеральном интернет экзамене, сводится к выбору набора заданий и вариантов ответов из уже существующей базы. Несмотря на простоту, такой метод является одним из самых распространенных в практическом применении. Разграничив задания по уровню сложности, и набрав большую базу заданий, преподаватель получает возможность быстро и уверенно проверять знания студентов по любому предмету и на любую тему, на которую есть задания. С другой стороны такой подход требует от преподавателя дополнительных трудозатрат на первоначальное заполнение базы данных заданий и вариантов ответов к ним. С таким вариантом также связана проблема списывания, ведь рано или поздно все возможные задания будут решены, и самые находчивые ученики будут иметь копию такой базы с вопросами и верными ответами к ним. Естественно подобная проблема довольно просто разрешима за счет увеличения количества заданий, вариантов ответов и уровней сложности заданий. Такой подход очень хорошо реализуется на практике проведения тестов, где есть определенное количество заданий на каждый вариант, они разделены по уровням сложности или тематике, и на каждый из вопросов можно указать один или более верных ответов. Обычно на решение подобных тестов отводится определенное количество времени, после которого ответы больше не принимаются.

Тут хочется отметить некоторые достоинства и недостатки методики обучения тестированием. Из достоинств можно отметить возможность проверки общих знаний учащихся по предмету, вместе с тем способность самих учащихся находить решения в условиях давления времени, т.е. когда учащийся заранее знает, что на решение задачи дано не более определенного промежутка времени, он подвергается психологическому воздействию, которое одних заставляет решать задачи быстрее, других наоборот – медленнее, на кого-то такое воздействие совсем не оказывает влияния. Так же в методике тестирования нет места для предвзятого отношения преподавателя к ученику, т.е. при выставлении оценки за тест она ставится исходя только из верности или не верности представленных ответов. Таким образом, тестирование позволяет проверить знания всех учащихся исходя из одного уровня сложности, результаты таких тестов обычно показывают преподавателю кто по знаниям выше, а кто ниже этого уровня. И дают возможность для дальнейшей индивидуальной работы. Из недостатков тестовой системы контроля знаний хотелось бы отметить только то, что такая система как раз не позволяет преподавателю подходить к каждому ученику индивидуально, ведь для одних такой тест будет слишком прост, для других – наоборот чересчур сложен. В результате и те и другие от подобного теста не получат ни необходимых знаний ни какой-либо должной их проверки.

Конечно, долгое время в России преобладала практика индивидуального подхода к обучению. Каждый преподаватель знал, на что способен определенный ученик, и не требовал от него большего или меньшего при проверке практических и самостоятельных заданий. Но такая практика постепенно изжила себя, поскольку учащихся становилось все больше, а значит, на одного преподавателя приходилось все больше забот и работы. Невозможно помнить уровень подготовки каждого ученика в потоке из 100-200 человек. Сейчас же не редкость, когда на одного преподавателя приходится в раз по 400-500 учащихся. При такой нагрузке естественно именно проверка тестированием является наиболее приемлемой. Однако тестирование не всегда подразумевает под собой выбор уже существующего ответа, зачастую в естественных науках, таких как физика, математика, химия в качестве тестов используются небольшие задания, решение которых не требует много времени, но позволяет проверить основные знания и навыки решения учащегося. Проблема использования таких заданий в том, что они, как правило, берутся преподавателем из уже готовых изданий и сборников задач. Таким образом, год от года набор этих заданий остается неизменным, и ученики уже знают, где можно найти ответы и даже готовые решения на любую задачу. Некоторые преподаватели самостоятельно придумывают задания для каждой новой проверки, однако такой подход наиболее трудоемок и совершенно неприемлем в современных учебных заведениях. Естественно, многие преподаватели уже не раз сталкивались с такой проблемой и у каждого, возможно, были свои идеи, как упростить проведение проверочных работ для себя, при этом, не понижая уровня самой проверки. Некоторые преподаватели, знакомые с языками программирования, самостоятельно создают простые приложения, реализующие алгоритмы генерации одного – двух типов задач.

Таким образом, может накопиться большое количество алгоритмов, реализованных на разных языках программирования, разными людьми и для разных целей. В результате может случиться так, что уже созданный ранее алгоритм будет утерян по невнимательности, или из-за отсутствия системы. То есть система генерации заданий должна не только позволять генерировать задания разных типов и алгоритмов, но и хранить эти алгоритмы, и систематизировать их по желанию пользователя.

## Обзор существующих решений по генерации задач.

На сегодняшний день очень часто можно услышать и прочитать про автоматизацию в совершенно разных сферах деятельности. Однако в сфере образования, наверно как одной из основополагающих, этот процесс идет крайне медленно и осторожно. И неудивительно, ведь неверный подход или действие могут привести к краху всей многолетней системы. Никто не возьмется сейчас нарушать сложившиеся годами традиции образования, однако, уже можно видеть небольшие шаги в направлении новой информационной эры. Так одним из наиболее важных нововведений в России за последние годы стало утверждение Федерального интернет экзамена (ФЭПО). Конечно, говорить об этом как о глобальном изменении пока рано, однако нет сомнений, что образование становится все более и более современным. Вместе с этим многие разработчики уже внесли свой вклад в развитие образования в информационной сфере. Большинство разработок, однако, не находят широкого применения, так как они изначально разработаны для узкой сферы применения, либо их функционал недостаточен, либо такое приложение не распространяется и его очень сложно получить. Все существующие сегодня решения можно разделить на несколько типов. В одних, как уже было отмечено выше, используется традиционный подход с выбором вариантов из набора существующих задач. Некоторые из них позволяют вводить свои варианты задач. Другие предлагают пользователю ограниченный набор типов задач или алгоритмов генерации с определенным набором параметров. В процессе исследования было найдено несколько подобных приложений. Все они разрабатывались в учебных заведениях преподавателями или студентами, однако, сами эти программы очень сложно достать, так как они распространяются только среди круга преподавателей этих вузов либо не распространяются вовсе.

Нет сомнений, что идея облегчить труд преподавателя не нова и исследуется уже многие годы. Однако, каких-либо серьезных нововведений кроме электронного тестирования до сих пор не последовало. Это означает лишь то, что этот вопрос остается нерешенным на сегодняшний момент. Либо решение этого вопроса остается за рамками общественного доступа. Во время исследования я нашел несколько описаний и решений в данном направлении. Почти все эти решения очень серьезны и охватывают огромную часть процесса обучения. Большинство из них используют принцип варьирования параметров в известной схеме, например в формуле. Этот принцип был реализован в работах Любченко В.В. «Поддержка процесса обучения с помощью системы поддержки принятия решений Combine» (математика), М.Н.Кирсанова «Генератор задач по теоретической механике и математике», В.В.Кручинина, Л.И.Магазинникова и Ю.В.Морозовой «Компьютерные самостоятельные работы по теории вероятностей», А.А.Притчина и А.А.Никитина «Программа-задачник по количественным соотношениям в неорганической химии», а также Л.Солдатовой и М.Сога «Автоматическая генерация задач для задач управления». На идеологии следующего уровня операциях с формулами основаны работы М.А.Левинской «Построение интеллектуальных обучающих систем по математике с использованием интеллектуальных обучающих систем с использованием языка TREE-REFAL» (варьирование формул в некотором «пространстве формул»), А.Ю.Ионова, И.А.Копылова и Д.О.Жукова «Генератор задач по физике» (соединение формул в цепочки по одинаковым переменным). Еще два метода рассматриваются в работе Л.А.Ашкинази и М.П.Гришкиной «Генератор задач по физике». Вместе во всех этих приложениях уже собраны множества задач и методов их генерации по самым разным предметам и направлениям, однако, отсутствие механизма распространения и расширения ведет за собой все новые работы в этой области, например эту, в которых поднимается та же проблема. Также большинство из рассмотренных разработок являются учебными работами, то есть их развитие, и распространение никак не ведется вне рамок отдельного учебного проекта. Тщательно рассмотрев все эти проблемы, было принято решение вынести вопрос распространения и развития программы на отдельный уровень.

## Распространение и развитие разработки.

Эта проблема оказалась гораздо важнее, чем может показаться. Многие решения остаются недоступными широким массам из-за отсутствия должного распространения. Немногие же из известных аналогов являются платными и давно перестали поддерживаться. Таким образом, в первую очередь встал вопрос о распространении и развитии данной разработки в дальнейшем. Наиболее верным и простым решением оказалось ведение разработки приложения в рамках лицензии распространения бесплатного программного обеспечения GNU GPL[[1]](#footnote-2). Эта лицензия подразумевает под собой множество пунктов, основа которых в том, что программный продукт, в рамках данной лицензии, является «свободным». Это означает, что кто угодно вправе использовать этот продукт и изменять его по своему желанию, с обязательным указанием на эти изменения при дальнейшем распространении. Также продукт, распространяемый в рамках данной лицензии и полученный таким путем не может быть использован в коммерческих целях, кроме отдельных случаев. Вообще на сегодняшний день уже существует новая лицензия GNU GPL v3, в которой еще жестче регламентируется использование продукта в коммерческих целях. Такой подход позволяет решить сразу несколько проблем. Во-первых, поскольку приложение распространяется бесплатно, оно будет более доступно и популярно, чем платное приложение (в этом правиле исключения, конечно, есть, но они касаются только продуктов мегакорпораций, на которые были потрачены огромные средства, и защиту которых в результате дешевле сломать, чем покупать лицензию). Во-вторых, разработка в рамках такой лицензии дает возможность в дальнейшем вовлечь в развитие проекта сторонних разработчиков. На сегодняшний день сотни программных продуктов разрабатываются в рамках подобных лицензий, многие пользователи даже не задумываются о том, сколько человек участвовало в разработке той или иной программы, и что они за это не получили ничего, кроме благодарностей коллег и знакомых. Для того чтобы проект можно было успешно развивать в рамках выбранной лицензии, его было решено зарегистрировать в интернете, на одном из сайтов специализирующихся на продвижении бесплатного и общедоступного программного обеспечения. Более подробно о том, что было сделано в этом направлении более подробно будет рассказано в описании практической части.

## Алгоритм, позволяющий параметризовать задачу.

При исследовании вопроса о создании генератора задач было рассмотрено множество идей по реализации подобного приложения. С математической точки зрения многие из них представляли собой набор правил, комбинируемых в разных вариантах и позволяющих создать задачу требуемой сложности. Полное и подробное описание этого метода нигде не приводится, поэтому это стало частью основной задачи проекта.

Для демонстрации возможностей программы оболочки было решено создать плагин, позволяющий генерировать задачи на основе таблиц параметризации. Использование таких таблиц лучше всего подходит для генерации простейших задач, требующих от одного – двух действий для решения. Как правило подобные задачи составляются преподавателями для закрепления материала (формул, теорем, законов) перед переходом к более сложным примерам. Давайте рассмотрим использование данного метода на примере. Опишем задачу нахождения координат точки, делящей отрезок в данном соотношении.

*Задача:*

*Точка M([R1]; [R2]) лежит на прямой, проходящей через две данные точки M1([R3]; [R4]), M2([R5]; R6]) и λ = M1M / MM2 = [R7] - отношение, в котором точка М делит отрезок М1М2. Найти [R].*

Где [R] из { M([R1]; [R2]), M1([R3]; [R4]), M2([R5]; [R6]), [R7] }

Здесь [R1] – [R7] это параметры задачи. Нарисуем небольшую таблицу связи, которая позволит нам быстрее понять принцип параметризации задачи.

Таблица 1. Соответствия параметров в задаче

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Параметр | Обозн. | min | max | Шаг. | Ед. изм. | Коэф. | Точн. | Связь |
| R1 | Координата Х точки М | x | -20 | 20 | 0,1 |  | 1 | 0,1 | (R3+R7\*R5)/(1+R7) |
| R2 | Координата Y точки М | y | -20 | 20 | 0,1 |  | 1 | 0,1 | (R4+R7\*R6)/(1+R7) |
| R3 | Коорд. Х1 точки М1 | X1 | -20 | 20 | 0,1 |  | 1 | 0,1 | R1\*(1+R7)-R7\*R5 |
| R4 | Коорд. Y1 точки М1 | Y1 | -20 | 20 | 0,1 |  | 1 | 0,1 | R2\*(1+R7)-R7\*R6 |
| R5 | Коорд. X2 точки М2 | X2 | -20 | 20 | 0,1 |  | 1 | 0,1 | (R1\*(1+R7)-R3)/R7 |
| R6 | Коорд. Y2 точки М2 | Y2 | -20 | 20 | 0,1 |  | 1 | 0,1 | (R2\*(1+R7)-R4)/R7 |
| R7 | Отношение M1M к MM2 | λ | -5 | 5 | 0,01 |  | 1 | 0,01 | (R3 – R1)/(R1-R5) |

Тут требуется дать несколько пояснений.

«Параметр» - это тот параметр который мы непосредственно выражаем в данной строке.

«Обозначение» - это его обозначение в формулировке задачи.

«Min» - это минимальное значение, которое может принимать данный параметр.

«Max» - это соответственно максимальное допустимое значение параметра.

«Шаг» - это требуемая точность параметра при подборе или генерации.

«Единица измерения» - это единица измерения параметра (в некоторых задачах параметр может измеряться в разных единицах).

«Коэффициент» - это число используемое при пересчете значения параметра из одних единиц в другие.

«Точность» - это та точность с которой идут вычисления в уравнении связи.

«Связь» - это формула показывающая зависимость параметра от значения других параметров, естественно сам параметр в явной форме не может участвовать в данной формуле.

Есть еще некоторые зависимости, которые можно было бы занести в нашу таблицу (например связь двух параметров в [R1] и [R2] как координат точки М), но считается что они незначительны для вычислений и генерации параметров.

Вот пример генерации задачи на основе такой таблицы:

*Точка M(1; 8) лежит на прямой, проходящей через две данные точки M1(3; 4,2), M2(Х2; Y2) и λ = M1M / MM2 = 2,5 - отношение, в котором точка М делит отрезок М1М2. Найти M2(Х2; Y2).*

*Ответ : M2(3; -9,5)*

Это очень простой пример, но он дает точное представление того, как работает данный алгоритм.

После составления подобной таблицы соответствия важно найти способ реализации, позволяющий быстро и наиболее точно генерировать задачи на основе таких таблиц. Одним из наиболее подходящих методов оказался метод градиентного спуска. Далее следует его краткое описание.

Пусть целевая функция имеет вид: .

И задача оптимизации задана следующим образом:

Основная идея метода заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом :

Где выбирается

* постоянной, в этом случае метод может расходиться;
* дробным шагом, т.е. длина шага в процессе спуска делится на некое число;
* наискорейшим спуском:

Для применения данного метода к таблицам параметризации необходимо определить ту самую целевую функцию f(x), которую необходимо минимизировать. Для этого положим g(xn) = Rn(X), где Rn(X) – уравнение связи n-ого параметра в таблице параметризации задачи (в нашем примере R1(X) = (R3+R7\*R5)/(1+R7)). Таким образом, возникает первое правило при составлении таблиц: все переменные в таблице должны быть из множества вещественных чисел . То есть, например, в таких таблицах нельзя использовать векторные переменные в явном виде а только в виде своих составляющих (вещественных координат). Далее необходимо определить еще один параметр xn = Rn, где Rn – искомый параметр. Сам R может быть выбран случайно, или же непосредственно указан как один из параметров в таблице. В нашем примере Rn = R1 = x (координата Х точки М). Таким образом можно показать, что необходимая нам функция f(x) = |g(xn) – xn| или, для более точных результатов, можно принять f(x) = (g(xn) – xn)2 эта функция неотрицательна для любых значений параметров и принимает нулевое значение в случае, когда g(xn) = xn. Другими словами, решив задачу минимизации для данной функции мы найдем все необходимые параметры таблицы. Здесь необходимо обозначить следующее правило составления таблицы: количество независимых связей должно быть равно количеству всех параметров. К связям относятся законы, формулы-описания и присвоения конкретных числовых значений. Это правило позволяет генерировать задачу для любого искомого параметра. В примере сгенерированной задачи, приведенном выше, в качестве искомого параметра на самом деле выступают два независимых параметра. Это возможно потому, что в их уравнениях связи они не указывают друг на друга. Для нашего случая функция f(x) = (g(x1)-x1)2 + (g(x2)-x2)2 сумма функций каждого независимого параметра. В этом случае так же выполняются все условия для нахождения параметров таблицы. Теперь можно вывести общую формулу целевой функции для n независимых параметров в таблице параметризации:

После того, как общий вид целевой функции задан, остается только реализовать метод градиентного спуска для данной функции. Более подробно реализация метода в рамках проекта показана в практической части.

Конечно, использование метода градиентного спуска может дать не точные результаты в ряде случаев, либо вообще не дать никаких результатов. Это связано с тем, что для более точных результатов целевую функцию необходимо подбирать под каждый конкретный пример. Либо можно иметь набор таких функций, позволяющий охватить большинство вариантов. Также на практике применяют различные вариации данного метода, например метод покоординатного спуска. Этот метод улучшает традиционный метод градиентного спуска за счёт того, что на очередной итерации спуск осуществляется постепенно вдоль каждой из координат, однако, теперь необходимо вычислять новые λ n раз за один шаг. Нет сомнений в том, что существует множество более точных и сложных методов, позволяющих решить поставленную задачу. Однако, целью этой работы не ставилось нахождение подобного метода, а только демонстрация возможностей работы основной программы оболочки.

## Реализация дополнительных алгоритмов генерации.

Конечно, рассмотрев только один единственный алгоритм очень трудно определить сложность и многогранность всех возможных алгоритмов связанных с генерацией задач. Именно здесь и начинает работать система плагинов. С ее помощью сторонний разработчик может без труда создать собственное приложение – дополнение для оболочки реализовав при этом свой собственный алгоритм или даже целую систему алгоритмов. Таким образом система предоставляет безграничные возможности для разработчиков и преподавателей работать совместно. Все описание алгоритма, его обоснование и какие-либо математические выкладки – все это забота преподавателей и ученых. Для программиста же при четко поставленной задаче, предоставленных материалах не составляет никакого труда написать нужное приложение. И конечно основное преимущество перед отдельной разработкой в данном случае будет то, что хранение и организацию информации уже производит программа – оболочка. И полученный в результате такой работы алгоритм очень легко передать коллегам. Естественно при определенных условиях можно создать плагин – хранилище, работающий с какой-либо базой данных и выбирающий оттуда заранее сформированные варианты заданий. Такой подход обсуждался выше и он конечно имеет право на существование. Нет сомнений, что при нормальном развитии проекта подобный плагин будет разработан одним из первых.

В результате свободное развитие проекта и открытость исходного кода раскроет множество уже решенных на сегодня задач для масс, и предоставит возможность для формирования более совершенных алгоритмов. Именно поэтому была поставлена задача как можно меньше ограничивать разработчика в будущем. Созданная система плагинов подразумевает под собой возможность разрабатывать полноценное приложение вне рамок проекта, таким образом, чтобы в дальнейшем интеграция с системой оболочкой не вызвала особых трудностей. А когда трудностей меньше – остальное дело техники.

## Программные средства.

На втором этапе выполнения задачи выбирались программные средства разработки. Основным критерием при выборе программных средств служили такие показатели как надежность, быстродействие, простота использования. В результате, основой всего проекта была выбрана программная платформа MicroSoft .NET (читается «дот нэт»). Выбор этой платформы для реализации проекта был связан с тем, что на сегодняшний день она является одной из самых современных и перспективных. Также данная платформа предлагает огромный набор готовых средств для быстрой разработки приложений.

Вместе с выбором программной платформы для разработки самой программы оболочки встал вопрос о реализации вывода текстов заданий для печати и электронного хранения. Здесь, естественно, был выбран язык разметки ТеХ. Его применение в оформлении и выводе различных научных статей и формул неоспоримо профессионалами на протяжении многих лет (первая версия ТеХ была разработана Д.Кнутом в 1989 году). Также на сегодняшний день существует огромное количество различных расширений и макросов для ядра системы. Рассказывать о языке ТеХ и его преимуществах можно бесконечно. Про этот язык написаны сотни книг и тысячи статей. В то время, как его ядро уже несколько лет не подвергалось изменениям, постоянно выходят новые наборы дополняющие и расширяющие функционал ТеХ-а в целом. Далее в практической части будет подробнее освещено как было реализовано взаимодействие программы оболочки и системы ТеХ.

Следующим шагом в выборе средств разработки был выбор варианта хранения данных. Главной целью нашей программы должно стать именно хранение и организация информации о задачах. Вместе с этим эта информация должна быть легко переносимой и легко изменяемой. Первоначально выбор был сделан в пользу СУБД, однако, из-за необходимости гибкого изменения и простой возможности переноса (например просто копированием) пришлось отказаться от нее в пользу простого хранения в файле. Конечно, у такого решения есть свои плюсы и минусы. Основным минусом является то, что при больших объемах информации поиск и извлечение информации из файла может осуществляться в несколько раз дольше чем при тех же объемах в СУБД. Однако, наша программа оболочка изначально не рассчитана на хранение и обработку информации промышленных масштабов. Применение программы сводится к персональному использованию, а значит, дополнительная нагрузка на ПК со стороны СУБД, в большинстве случаев, наоборот может замедлить работу. В дополнение к этому применение СУБД влечет за собой дополнительные проблемы при переносе данных.

## Программная платформа .NET особенности, архитектура.

Летом на конференции Professional Developer's Conference 2000 Microsoft представила миру платформу .NET, которая предлагает новый путь разработки программного обеспечения. .NET – это платформа для построения интегрированных, сервис-ориентированных приложений, отвечающих современным потребностям Интернет коммерции. Она позволяет быстро строить Web сервисы и приложения, оперирующие информацией из многих источников, и независящие от платформ и языков программирования. Платформа .NET не привязана к операционной системе Windows и впоследствии может быть реализована для других операционных систем.

Платформа .NET упрощает разработку приложений и повышает надежность кода. В частности, она обеспечивает автоматическое управление временем жизни объектов, нейтральные к языкам библиотеки классов и пересекающие границы языков наследование, обработку исключений и отладку.

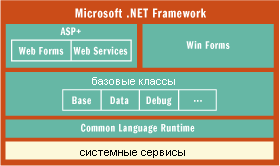


Рис 1. Архитектура .NET (рисунок взят с сайта Microsoft)

«Common Language Runtime» (CLR, с англ. «Общеязыковая среда исполнения») опирается на системные сервисы операционной системы и управляет выполнением кода, написанного на любом современном языке программирования. Набор базовых классов дает доступ к сервисам платформы, которые разработчики могут использовать из любого языка программирования. «Common Language Runtime» и базовые классы вместе составляют основу .NET платформы.

CLR предоставляет исполняемому коду определенный набор сервисов. Например, CLR поддерживает создание и манипулирование потоками. Поэтому любой язык, который может использовать CLR, сможет использовать потоки. Код, который нуждается в CLR во время выполнения, называется “управляемый” (managed) код. Ответственность за такие задачи как создание объектов и вызов методов возлагается на CLR. Код, который не нуждается в CLR, называется “неуправляемый” (unmanaged) код.

Microsoft предоставляет 4 компилятора, которые генерируют код для .NET CLR: C++, C#, Visual Basic (включая VBScript и Visual Basic for Applications) и JScript. Visual C++ - это единственный компилятор, который может генерировать неуправляемый код. Остальные компиляторы могут производить только управляемый код, и поэтому код, написанный на этих языках, всегда нуждается в CLR.

Управляемый код – новое понятие, которое поначалу вызывает затруднения для понимания у большинства программистов. Его особенностью является то, что он выполняется под постоянным надзором среды исполнения, которая контролирует все его шаги. Именно из-за столь сильного контроля и надзора было решено назвать код управляемым.

Когда файлы с исходными текстами готовы, вы запускаете компилятор и получаете EXE или DLL. Эти EXE или DLL файлы очень похожи на PE (Portable Executable – портируемые выполняемые) файлы, по сути дела они и есть PE файлы с некоторыми отличиями.

Первое отличие заключается в том, что код в управляемых PE файлах не является командами процессора x86 или другим машинным кодом. Вместо этого компилятор создает код на Промежуточном Языке Microsoft (Microsoft intermediate language - MSIL). PE файл, содержащий MSIL может выполняться на платформе любого процессора, если операционная система, предоставляет .NET CLR.

Второе отличие заключается в том, что этот файл содержит метаданные, которые используются CLR для обнаружения и загрузки типов из файла, расположения объектов в памяти, вызова методов, управления ссылками, перевода MSIL в машинные коды, контроля за безопасностью и множества других задач.

Еще одно отличие – то что полученные компоненты не просто EXE или DLL файлы. Единица использования и размещения в .NET – это «assembly». В зависимости от опций компилятора можно получить однофайловый «assembly» или модуль с управляемым кодом, распространяемый как часть многофайлового «assembly». С точки зрения клиента, «assembly» – это именованная коллекция экспортируемых типов и ресурсов. С точки же зрения разработчика «assembly», – это коллекция PE файлов, файлов ресурсов, phtmlL страниц, картинок и т.п.

MSIL – это процессоронезависимый промежуточный язык, созданный Microsoft. MSIL - язык более высокого уровня, чем большинство машинных языков. Он понимает типы объектов и имеет инструкции для создания и инициализации объектов, вызова виртуальных методы и непосредственной манипуляции элементами массива. Он даже имеет инструкции, которые оперируют исключениями. Как и любой другой машинный язык, MSIL может быть написан на ассемблере. Microsoft предоставляет ассемблер и дизассемблер для MSIL.

Перед тем, как выполнять управляемый код CLR должен сначала скомпилировать управляемые MSIL инструкции в инструкции процессора. Здесь возникает типичная проблема: когда пользователь запускает программу, он не намерен ждать пока вся программа скомпилируется, тем более, что большинство функций программы не будут вызваны. Поэтому CLR, компилирует MSIL код в инструкции процессора, когда функции непосредственно вызваны. Всякий раз, когда такая функция вызывается в будущем, сразу выполняется машинный код (а компилятор уже не вовлекается в процесс). Поскольку MSIL компилируется только в нужный момент (just-in-time - JIT), этот компонент CLR часто упоминается как JIT компилятор (JIT compiler) или «JITter».

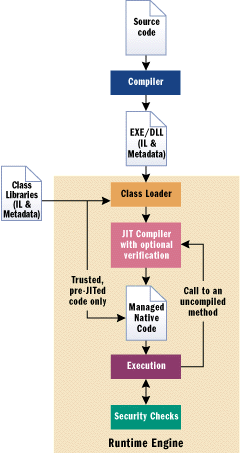


Рис 2. Жизненный цикл сборки (картинка с сайта Microsoft).

(Перевод Рис 2: исходный код - компилятор - exe/dll (IL и метаданные) – загрузчик класса - JIT компилятор с необязательной проверкой - машинные коды - выполнение - проверки безопасности; Библиотеки классов (IL и метаданные); безопасный прекомпилированный код; вызов некомпилированного метода)

Программисты на низкоуровневых языках типа C или C++ возможно думают об эффективности выполнения описанной схемы. Ведь неуправляемый код компилируется сразу в инструкции процессора и при вызове просто выполняется. А в управляемом окружении чтобы выполнить код, MSIL должен компилироваться в инструкции процессора в реальном времени, потребляя большое количество памяти и мощности процессора.

Действительно, управляемый код выполняется медленнее и имеет больше накладных расходов, чем неуправляемый код. Но Microsoft предлагает инструмент, позволяющий выполнять полную компиляцию «assembly» в машинные коды и сохранять результат на диске. Когда «assembly» загрузится в следующий раз, будет использована эта сохраненная версия и приложение стартует быстрее.

Отметим особо, что в приложении можно совмещать фрагменты управляемого и неуправляемого кода. Неуправляемый код следует использовать для критичных к времени выполнения функций.

Над CLR в архитектуре .NET находится инфраструктура сервисов. Эта инфраструктура предоставляет классы, которые могут быть использованы из любого языка программирования. Каждый класс дает доступ к некоторому элементу основной платформы.

Для примера приведем несколько пространств имен («namespaces») и их классы.

Таблица . Пространства имен.

|  |  |  |
| --- | --- | --- |
| **Пространство имен** | **Назначение содержащихся классов** | **Пример классов** |
| System | Реализация типов, используемых каждым приложением | Object, Byte, Array, Int32, Exception, String |
| System.Collections | Управление наборами данных | ArrayList, Dictionary, Hashtable, Queue |
| System.Data | Работа с базами данных | DataBinding, DataTable, DataSource |
| System.IO | Чтение и запись | ByteStream, File, FileStream, MemoryStream |
| System.Net | Сетевые взаимодействия | WebRequest, UdpClient, Sockets |
| System.Web.UI.WebControls | Построение пользовательского интерфейса для приложений, ориентированных на Web | DataGrid, HyperLink, ListBox, RadioButton, Table |
| System.WinForms | Построение пользовательского интерфейса для локальных приложений | Button, CheckBox, DataGrid, FileDialog, ListBox, MainMenu |

Имеются также классы для рисования (GDI+), работы с потоками, национальной поддержки, криптографии, сериализации, и т.д.

Также есть, например, классы, предназначенные для инструментов разработки и позволяющие выполнять такие функции как отладка-трассировка, создание ресурсов, конфигурирование-инсталляция, получение лога событий и оценка эффективности. Методы класса могут быть перегружены, поэтому методы, отличающиеся лишь немного по поведению, имеют идентичные имена и различается только прототипами.

.NET - полностью объектно-ориентированная платформа. Программисты могут создавать собственные пространства имен, содержащие их собственные классы. Это значительно упрощает разработку программного обеспечения по сравнению с классическими Windows парадигмами программирования. Так как все услуги платформы предлагаются через объектно-ориентированную парадигму, разработчики должны иметь некоторое понимание объектно-ориентированного программирования.

## Преимущества и недостатки новой платформы в разработке.

Программная платформа .NET совместно с современными средствами разработки предоставляет программисту практически безграничные возможности для создания приложений любого уровня.

Перед разработчиками платформы .NET была поставлена весьма нетриви­альная задача — обеспечить полную межъязыковую интеграцию в рамках среды. Ранее подобные цели ставились перед разработчиками СОМ (Com­ponent Object Model, Компонентная Объектная Модель), но им не удалось добиться желаемого результата. Формально, конечно, СОМ предоставляет все необходимые для межъязыковой интеграции сервисы. Свидетельством этого является возможность организации взаимодействия с платформой СОМ из различных языков, начиная от Ассемблера и заканчивая простей­шими языками сценариев вроде VBScript. Однако, с пользовательской точки зрения, взаимодействие было организовано настолько неудобно и запутанно, что его вообще мало кто понимал. Главный просчёт разработчиков СОМ состоял в том, что основные заботы по межъязыковой интеграции были возложены на рядовых программистов. Именно они должны были сглажи­вать различия между языками, путём введения унифицированных интерфей­сов, информация о которых предоставлялась через библиотеки типов.

Разработчики .NET решили пойти другим путём — они ввели систему мета­данных, описывающих не только внешние интерфейсы, предоставляемые приложениями и компонентами, но также их полную внутреннюю структу­ру. Причем метаданные спроектированы как неотъемлемая часть любого управляемого приложения. Теперь больше не надо создавать дополнитель­ные заголовочные файлы или внешние библиотеки типов. Вся информация предоставляется самими приложениями и хранится совместно с их кодом. Благодаря этому, стало возможным взаимодействие как через внешние, спе­циально предоставленные интерфейсы, так и через прямое обращение к внут­ренним программным элементам. Теперь можно использовать типы других приложений как свои собственные — применять в качестве базовых, созда­вать их экземпляры, обращаться к их методам и т. д. Обобщая, можно ска­зать, что метаданные — это новый виток в эволюции технологий предостав­ления информации о типах и программах в целом.

Подобная модель взаимодействия существенно упрощает жизнь рядовым программистам, потому что теперь им не нужно задумываться об описании внешних интерфейсов, которые не так уж и просты для понимания. Мно­гим начинающим программистам совершенно непонятно, для чего, напри­мер, нужны заголовочные файлы в C/C++ подобных языках, или еще ху­же — зачем необходимо создавать библиотеки типов для СОМ-компонентов. Теперь рядовые программисты отстранены от необходимости решения вопро­са, метаданные автоматически предоставят необходимую информацию как о внешних, так и о внутренних интерфейсах.

Все современные компиляторы, совместимые с платформой .NET, генери­руют из текста программы не только исполняемый код, но и метаданные, описывающие программную модель приложения. Взаимодействие же приложений осуществляется не на уровне исходных ко­дов, а на уровне, чётко стандартизированных универсальных метаданных.

Фактически, метаданные являются унифицированной и чётко стандартизи­рованной технологией описания исходного кода. Они являются выжимкой той, действительно необходимой информации, которая содержится в перво­зданном исходном коде приложения. Среда исполнения имеет полное представление о внутреннем устройстве программы, которое она по­лучает из метаданных. Она знает, как устроена программа, и может предос­тавить сервисы для взаимодействия с другими программами. Причем эти сервисы работают полностью в автоматическом режиме, абсолютно про­зрачно для программиста.По сути дела, на уровне самой среды исполнения никаких языков и нет, там есть лишь мета­данные и IL. Сама среда исполнения даже представления не имеет, на каком языке изначально было написано приложение. Точно так же, как процессор не имеет представления о языке, на котором разрабатывалась программа. Эта информация не нужна для исполнения приложений. На сегодняшний день существует огромный перечень языков поддерживающих технологию .NET.

Встроенные (поставляются вместе с .NET Framework):

* C#
* J#
* VB.NET
* JScript .NET
* C++/CLI — новая версия C++ (Managed)

Поддерживающие .NET не активно:

* [Ada](http://ru.wikipedia.org/wiki/Ada) — см. [A#](http://ru.wikipedia.org/w/index.php?title=A_Sharp&action=edit&redlink=1)
* APL
* [Boo](http://ru.wikipedia.org/wiki/Boo), основан на [Python](http://ru.wikipedia.org/wiki/Python)
* [COBOL](http://ru.wikipedia.org/wiki/COBOL)
* [Component Pascal](http://ru.wikipedia.org/wiki/Component_Pascal) (Component Pascal ближе к [Oberon](http://ru.wikipedia.org/wiki/Oberon) чем к [Pascal](http://ru.wikipedia.org/wiki/Pascal))
* [Delphi](http://ru.wikipedia.org/wiki/Delphi): Delphi 8, Delphi 2005, Delphi 2006
* [Eiffel](http://ru.wikipedia.org/wiki/%D0%AD%D0%B9%D1%84%D0%B5%D0%BB%D1%8C_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29)
* [F#](http://ru.wikipedia.org/wiki/F_Sharp), член семейства языков программирования [ML](http://ru.wikipedia.org/wiki/ML)
* [Forth](http://ru.wikipedia.org/wiki/Forth_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29)
* [FORTRAN](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D1%82%D1%80%D0%B0%D0%BD)
* [Haskell](http://ru.wikipedia.org/wiki/Haskell)
* [IronPython](http://ru.wikipedia.org/wiki/IronPython) — реализация языка [Python](http://ru.wikipedia.org/wiki/Python)
* [Lexico](http://ru.wikipedia.org/w/index.php?title=Lexico&action=edit&redlink=1)
* [Lisp](http://ru.wikipedia.org/wiki/Lisp)
* [Mercury](http://ru.wikipedia.org/wiki/Mercury_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29)
* [Mondrian](http://ru.wikipedia.org/w/index.php?title=Mondrian&action=edit&redlink=1)
* [Nemerle](http://ru.wikipedia.org/wiki/Nemerle) — гибридный функционально/императивный язык
* [Oberon](http://ru.wikipedia.org/wiki/Oberon)/[Zonnon](http://ru.wikipedia.org/wiki/Zonnon)
* [Perl](http://ru.wikipedia.org/wiki/Perl)
* [PHP](http://ru.wikipedia.org/wiki/PHP)
* [RPG](http://ru.wikipedia.org/wiki/RPG_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29)
* [Ruby](http://ru.wikipedia.org/wiki/Ruby)
* [Smalltalk](http://ru.wikipedia.org/wiki/Smalltalk)

Из всего вышесказанного можно сделать несколько очень важных выводов. Во-первых, разработка приложения на основе платформы .NET представляет собой непрерывный процесс взаимодействия программиста среды исполнения CLR. Основное преимущество такого подхода – это отсутствие ограничений на среду и язык разработки. Если вы умеете программировать на каком-либо языке программирования, не составляет труда узнать существует ли его реализация (обычно это просто другая версия компилятора) для платформы .NET. Если да, то вам ничего не придется переучивать и все ваши алгоритмы можно будет использовать в готовом виде. Если же нужный вам язык программирования еще не был перенесен на платформу .NET, то у вас есть огромный выбор языков для изучения – от традиционного C#, который был разработан специально для новой платформы, до экзотических языков вроде F# или Ada. Именно это можно назвать недостатком новой платформы – необходимость изучать новые языки программирования. В таком случае всегда рекомендуют изучать язык C#, как самый популярный и самый развивающийся в этой среде. Зная основы ООП на изучение базовых принципов и структур нового языка уйдет совсем немного времени. Тем более о нем написаны сотни книг.

С# вобрал в себя все лучшее из таких популярных языков как С++, Visual Basic, Java и Object Pascal. С# обеспечивает быструю разработку, в то же время позволяет писать эффективный код. Перечислим особенности нового языка:

* автоматическая уборка мусора
* возможность манипулировать указателями и иметь непосредственный доступ к    памяти
* поддержка свойств и событий (аналогично VB)
* поддержка атрибутов
* встроенная поддержка основных типов (строка, массив, ...)
* множественное наследование возможно только от интерфейсов (как в Java)
* поддержка С API, Windows API и COM+ на уровне языка
* поддержка «assembly»
* контроль типов
* автоматическая инициализация переменных

Стоит отметить, что языком для реализации программы оболочки был выбран именно C#.

Второй очень важный вывод – это безболезненная интеграция. Как было отмечено библиотека написанная на любом языке для платформы .NET становится доступной в простой и удобной форме для использования в любом другом языке поддерживающем .NET. Именно это позволило создать такое приложение, которое бы интегрировалось с множеством других при помощи простого копирования. Именно наличие метаданных внутри каждой отдельной библиотеки позволило создать столь гибкую систему плагинов.

Наконец третий вывод касается скорее недостатка в использовании новой платформы. Он заключается в принципе работы исполняемых фалов. А именно в использовании промежуточной среды CLR. Самым большим недостатком тут становится необходимость установки этой среды у пользователя. Во всех последних выпусках своей операционной системы, начиная с Windows XP, корпорация Microsoft позаботилась о наличии интегрированной поддержки платформы .NET. Однако если вы планируете более широкое распространение своих разработок, вам необходимо предусмотреть установку платформы .NET на персональный компьютер пользователя.

До недавнего времени реализация среды CLR была доступна только под операционную систему MS Windows®, однако, вот уже несколько лет успешно развивается при поддержке компании Novell, и признан компанией Microsoft официально, проект Mono, реализующий платформу .NET на базе свободного программного обеспечения. Уже сегодня доступны реализации этого проекта для операционных систем [GNU/Linux](http://ru.wikipedia.org/wiki/GNU/Linux), [Mac OS X](http://ru.wikipedia.org/wiki/Mac_OS_X), [Solaris](http://ru.wikipedia.org/wiki/Solaris_%28%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%29) и [Unix](http://ru.wikipedia.org/wiki/Unix). Таким образом, уже сейчас можно называть любой проект, реализованный на платформе .NET – кроссплатформенным. И уже этот факт можно считать преимуществом новой системы.

# Практическая часть.

## Для кого эта программа.

Как уже отмечалось в начале этой работы, программа разработана для использования теми, кто занимается научно-преподавательской деятельностью. Спектр ее применения невероятно широк, от простых тестовых заданий, до зачетов и экзаменов. Сочетая в себе современную платформу и простой, дружественный интерфейс, программа позволяет подобрать набор нужных заданий в нужном количестве, сгенерировать в необходимой форме, и, при необходимости, сохранить этот набор для дальнейшего использования. Для более опытных преподавателей, знакомых с языками программирования, эта система может стать незаменимым помощником при организации и использовании собственных алгоритмов. Поскольку сама система распространяется свободно, ее можно с легкостью использовать на практических занятиях. Можно давать учащимся в качестве промежуточной проверки разработку нового дополнения, основанного на пройденном материале. Такой подход, несомненно, положительно скажется на всем учебном процессе. Однако необходимость использования этой программы невозможно навязывать. У каждого преподавателя свои, годами складывающиеся, взгляды на процесс обучения. Одни готовы к нововведениям в силу своей открытости или в силу того, что совсем недавно начали преподавать. Другие, напротив, предпочитают традиционные для себя методы, которыми они пользовались на протяжении долгих лет. И все же польза от применения этой программы может быть у всех, ведь если преподаватель не умеет программировать, но хорошо владеет теорией, он ожжет с легкостью предоставить несколько алгоритмов для новых дополнений. Позже, увидев результат своей работы, преподавателю может понравиться или он увидит недочеты в самом алгоритме, и тогда нужно уже браться за исследовательскую работу. Именно к этому и подталкивает любого преподавателя данная программа оболочка. Она не может существовать без плагинов – дополнений, а разработка алгоритмов, требующая огромных знаний предмета, является основой любого дополнения. Конечно, очень много алгоритмов известно уже сейчас, но многие из них применяются только в теории. Нам же хочется, чтобы исследователь чувствовали необходимость их труда. Придумывая и открывая все новые теории и законы, нужно их применять на практике.

Вместе со всем вышесказанным, использование данной программы только преподавателями не раскрывает весь ее потенциал. Основа системы обучения – это сами учащиеся. Без них, конечно, не было бы никакого процесса. И для учащихся эта программа представляет не меньшую пользу, чем для их учителей. Во-первых, как уже было сказано, разработка собственных дополнений для данной программы может служить отличной проверкой знаний не только по профильному предмету, но и по программированию, если таковое включено в программу обучения. В идеале, можно объединять учащихся с разных профилей, чтобы затем они могли совместно придумать и воплотить какой-либо новый алгоритм. В этом случае, как учащиеся, так и преподаватели получат максимум отдачи от процесса обучения, ведь для разработки чего-то нового необходимо глубже изучить предмет, проявить творческий подход и грамотную работу в команде. Также можно при разработке нового дополнения требовать грамотно оформленного обоснования, почему их алгоритм уникален и чем он превосходит другие. Конечно, более сложные и серьезные алгоритмы могут быть оформлены в виде курсовых или даже дипломных работ.

С другой стороны учащиеся смогут использовать эту программу для самоподготовки. Во время прохождения определенной темы, если уже имеются нужные дополнения, можно будет с легкостью приготовить для себя самого домашнюю самостоятельную работу. Это позволит ускорить процесс обучения во время занятий с преподавателем.

Как мы видим применение данной разработки возможно в различных ситуациях. Не исключено, что может обнаружиться масса других способов улучшения процесса обучения при помощи этой программы.

## План разработки. Выбор средств.

После описания целей и задач данного проекта последовало составление плана разработки самого приложения. Фактически этот план был основан на нескольких пунктах:

1. Анализ существующих решений.
2. Разработка идеи работы приложения.
3. Составление модели, архитектуры приложения.
4. Выбор способа хранения данных.
5. Выбор способа отображения данных.
6. Выбор языка программирования.
7. Собственно разработка основной программы оболочки.

После этого последовал шаг решения вопроса о распространении программы и разработка собственного дополнения, о котором было рассказано раньше. Пройдем по пунктам.

Первым и главным шагом стал, конечно, поиск и анализ подобных решений. При первоначальном поиске выяснилось, что все решения, так или иначе затрагивающие выбранную тему, очень сложно найти в виде готового программного продукта. Обнаружилось несколько коммерческих продуктов, среди которых лидирующее место занимает «Computer mentor» (компьютерный наставник) – разработка российской фирмы, однако его платность и ограниченность заданиями только для старших классов школы дала толчок новым поискам. После этого были найдены описания для нескольких программ реализующих генерацию задач по определенным параметрам. В основном, конечно, это были задачи по физике и математике. Сами же программы найти так и не удалось. Таким образом, было решено продолжать именно в направлении расширяемости и общедоступности приложения.

Второй шаг плавно последовал из первого. Исследовав несколько коммерческих решений и просмотрев описания других систем, было решено создавать приложение, основой которого станет не какой-то определенный алгоритм или задача, а именно возможность добавлять эти алгоритмы самостоятельно. Вместе с этим возникла необходимость хранить и организовывать эти дополнения.

Третьим шагом стала разработка архитектуры и принципов будущего приложения. На этом этапе уже было решено, что программа будет представлять собой систему плагинов, добавляемых простым копированием. Таким образом, оставалось только разработать рабочую архитектуру с учетом всевозможных требований в будущем.

Рис 3. Архитектура приложения.

В общих чертах на рис 3 изображена архитектура разрабатываемого приложения. Эта упрощенная модель позволяет увидеть главное – само приложение (ядро) не должно быть привязано к какому-либо дополнению. Также данная модель позволяет абстрагироваться от конкретной модели хранения данных. Это было сделано для того, чтобы можно было легко изменить тип хранилища в будущем. Так же можно увидеть, что модуль печати вынесен отдельно. Это сделано по тем же причинам. О модуле печати будет рассказано далее. Таким образом, основными принципами работы приложения мы выделили универсальность и расширяемость.

На четвертом этапе выбиралась реализация хранилища данных. Было проведено небольшое исследование, которое показало, что применение сложных систем вроде СУБД в данном случае может повредить всей системе в целом. Связано это с тем, что изначально объемы информации, предполагаемой для хранения, очень малы, и установка отдельного сервера СУБД может замедлить работу системы не только при работе с самим приложением, но и при других обстоятельствах. По этим причинам было решено реализовать хранение данных в обычном файле. Сам файл представляет собой зашифрованный набор данных, получаемый при помощи метода сериализации (перевода программного объекта в его хранимое представление). Далее при описании системы отдельное внимание будет уделено тому, какие преимущества получает система в случае файлового хранилища.

Следующий этап разработки был, пожалуй, самым ответственным. На этом этапе предстояло выбрать средство для отображения сгенерированных данных для пользователя. При этом нельзя было забывать о том, что как дополнения генераторы, так и сам модуль печати должны быть независимы друг от друга и должны правильно взаимодействовать посредством ядра программы. Для этого было решено свести информацию для вывода к одному унифицированному формату, из которого в дальнейшем можно будет выводить любую информацию для пользователя. Первым возможным вариантом был простой текст, однако, он практически сразу нам не подошел, так как простой текст не позволяет провести простое форматирование (жирный шрифт, увеличенный шрифт). Сразу после этого встал вопрос о выводе различных формул, сложность которых определяется только сложностью всего алгоритма генерации и типом задачи. Первым вариантом была разработка собственного формата обмена данными основанного на XML (англ. «eXtensible Markup Language» — расширяемый язык разметки; произносится [экс-эм-э́л])[[2]](#footnote-3). Такое решение показалось верным, однако, когда начался разбор всевозможных вариантов оформления и типов формул, было ясно, что разработка такого формата займет длительное время. Язык разметки ТеХ рассматривался нами с самого начала, однако, передавать данные только при помощи этого языка было неверным, после исследований и попытки создать формат обмена на XML, было принято решение перевести формат языка ТеХ в рамки формата XML. В результате небольшого поиска обнаружилось, что уже существует подобный формат[[3]](#footnote-4), и даже разработано приложение, позволяющее переводить данные из формата XML в формат LaTeX (одна из самых популярных разновидностей ТеХ). Называется этот формат TeXML, его разработка ведется в рамках свободной лицензии, которая позволяет использовать его в любых целях. В рамках проекта была разработана специальная библиотека, упрощающая работу программиста с данным форматом. Описание этой библиотеки и ее функций прилагается к программе. Так же на сайте разработчика находится полная спецификация по использованию данного формата на английском языке. Эта спецификация также приведена в приложении к данной работе.

Таким образом, решение о формате передачи данных и выводе их для пользователя свелось к использованию языка разметки ТеХ. Далее нам необходимо было определить методику работы модуля печати. Его реализация оказалась сложна тем, что было необходимо использовать весь пакет ТеХ для корректного вывода информации в печатную форму. Для этих целей мы решили использовать популярный в интернете набор MiKTeX, который позволяет создавать собственные наборы модулей TeX. В качестве основного интерпретатора был выбран модуль LaTeX, так как именно в его формат преобразует данные из XML используемая нами программа TeXML. Также в состав сборки были включены все самые популярные пакеты и шрифты. Одна из главных сложностей заключалась в том, что ядро MiKTeX привязано к операционной системе, и требует установки с участием пользователя. Для решения этой проблемы из уже установленной сборки была создана «переносная» сборка, установка которой уже не требует вмешательства пользователя и выполняется при первом запуске системы автоматически. Конечно, и здесь мы предусмотрели некоторую гибкость системы. Несмотря на то, что мы постарались выбрать все наиболее популярные пакеты. Всех пожеланий учесть невозможно, а полный набор пакетов занимает слишком много места на жестком диске, при этом существуют несовместимые пакеты. Для того чтобы можно было изменить набор пакетов в сборке MiKTeX предоставляется стандартная программа с понятным интерфейсом и принцип ее использования будет понятен каждому, все что требуется от пользователя – это выбрать недостающие пакеты из списка и программа автоматически установит их через интернет либо с локального носителя. Вывод полученной информации осуществляется в формат PDF (Portable Document Format), хорошо известный и подходящий для печати форматированного текста.

После проведения всех подготовительных работ, последовал этап выбора основного языка программирования. Как было отмечено ранее, в качестве основного языка программирования был выбран язык C#, основанный на платформе .NET. В первую очередь было необходимо найти язык широкого применения, то есть позволяющий производить сложные операции с минимумом усилий. Во-вторых, было понятно, что в рамках этого проекта нельзя говорить о разработке на каком-то конкретном языке. По своей сути разработанная система дополнений подразумевает возможность разработки на нескольких языках. Таким образом, необходимо было выбрать некоторое семейство языков, связанных и способных взаимодействовать с максимальной легкостью. По всем этим критериям подошел именно язык C#, вместе с другими языками, реализующими платформу .NET, он представляет собой идеальное средство разработки динамической системы. В дополнение к этому данный язык программирования хорошо знаком автору работы.

Последним этапом в процессе создания приложения стала непосредственная его разработка при помощи выбранных средств. В связи с тщательной подготовкой, времени на этот этап осталось очень мало, поэтому в конечном варианте все еще имеются недочеты, но их исправление лишь дело времени.

## Вопросы расширяемости и распространения системы.

Уже на ранних этапах разработки возник вопрос о распространении и дальнейшем развитии приложения. Для того чтобы в дальнейшем было максимально просто получить доступ к самому приложению и принять участие в его развитии, приложение было опубликовано под свободной лицензией GNU GPL и размещено вместе с исходным кодом в интернете[[4]](#footnote-5) на специализированных сайтах. Для точности укажу, что основной страницей проекта в интернете является проект в рамках программы «Google Code», он размещен по адресу http://code.google.com/p/dappgnr/. На этой странице можно найти краткое описание системы, материалы по ней, а также принять участие в разработке самой системы. Такой подход к разработке и развитию приложения уже зарекомендовал себя как перспективный для любого проекта с ограниченным бюджетом. Сам текст лицензии на английском языке прикреплен в Приложении 2. Предоставление данной лицензии в оригинале вместе с распространяемым приложением и сопутствующей документацией является одним из основных ее требований. Неофициальный перевод лицензии на русский язык можно найти в интернете по адресу http://code.google.com/p/gpl3rus/wiki/LatestRelease.

Проблема расширяемости приложения за счет дополнительных алгоритмов лежит в основе всей работы, поэтому ее решение является столь важным. Первой задачей в этом направлении стала собственно разработка системы плагинов. Ее работа основана на простом принципе, используемом в большинстве подобных систем: специально для системы создана библиотека «интерфейсов» (в рамках программирования это фактически описание наборов функций), наследуя эти интерфейсы, программист со своей стороны решает несколько важных задач. Во-первых, он не должен больше заботиться о совместимости с основной программой, так как программа взаимодействует с плагином только через описанные в «интерфейсе» функции. Во-вторых, программист точно знает назначение каждой функции, и в какой момент в программе происходит ее вызов. На самом деле, достаточно только назначения. Вся библиотека подробно описана в Приложении 1. В эту библиотеку входят «интерфейсы» для дополнений – генераторов, а также и для дополнений – принтеров.

Разработка собственных вариантов вывода на печать или просто в файл была заложена как одна из основных. Такой подход позволит разработчикам создавать специализированные библиотеки печати (например, в MS Word, HTML или другие форматы). Без сомнения это очень хорошо отразиться на общем развитии программы.

## Описание системы.

При проектировании системы особое внимание было уделено необходимой динамичности программы, для этого была создана структура плагинов (прикладных библиотек), расширяющих базовый функционал системы. По своей сути сама система представляет собой оболочку для реализации связи между пользователем и функционалом плагина. Также система позволяет хранить и организовывать информацию о задачах – генераторах.

Система состоит из нескольких взаимосвязанных частей. При первом запуске система автоматически регистрирует компоненты ТеХ для корректной работы модуля печати (Рис 4).

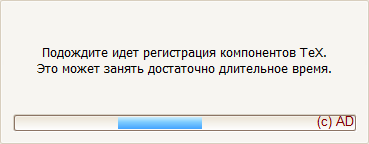


Рис 4. Инициализация системы.

Далее запускается главная форма (Рис 5), здесь пользователь сразу видит список объектов, позволяющих генерировать задачи (назовем их «Задачи»).

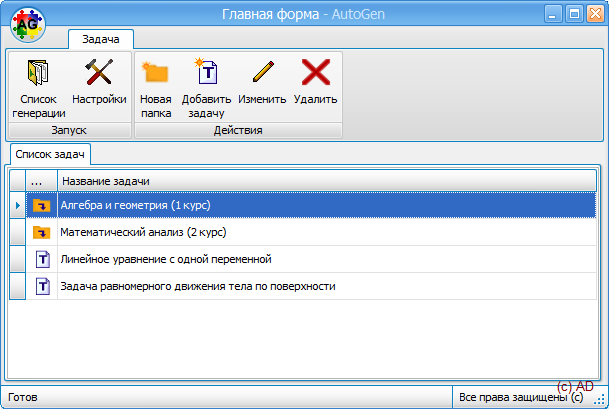


Рис 5. Главная форма программы.

Данный список поддерживает структурирование папками, с неограниченной вложенностью. Это позволит расположить все созданные задания таким образом, чтобы пользователь сам мог быстро найти нужное ему. В панели инструментов находятся инструменты, позволяющие создавать и изменять структуру папок, а также создавать задания на основе загруженных плагинов. При нажатии на кнопку «Добавить задачу» открывается диалог, предлагающий выбрать плагин на основе которого будет создана Задача и название для отображения Задачи в списке (Рис 6). Естественно из одного плагина может быть создано несколько экземпляров Задач.

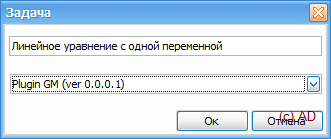


Рис 6. Добавление новой задачи.

После добавления Задачи, она отображается в списке и можно начинать редактировать ее параметры. Изменение происходит на вновь созданной вкладке (Рис 7), это позволяет максимально быстро ориентироваться в приложении и не засоряет рабочий стол большим количеством окон. Содержимое вкладки зависит от конкретной Задачи и плагина, с помощью которого была создана эта Задача.

В нашем плагине для редактирования задачи доступно «Имя задачи» и «Автор». Далее пользователь заполняет таблицу параметризации. На внутренней вкладке «Печать» необходимо ввести шаблон текста для данной задачи с использованием специальных символов (Рис 8). В нашем примере названия переменных находятся между сочетаниями символов «{$» и «#}» это сделано для того, чтобы выделить их в тексте программы. Этот же синтаксис используется при описании уравнений связи. Сочетания символов можно изменить на внутренней вкладке «Печать». Здесь же находятся названия специальных переменных, позволяющих пользователю вставить в текст шаблона значение искомого параметра («Результат») и его представление («Имя искомой переменной»). В этом же шаблоне он может использовать любые команды языка ТеХ, весь текст передается практически без изменений, добавляется только заголовочная часть документа.

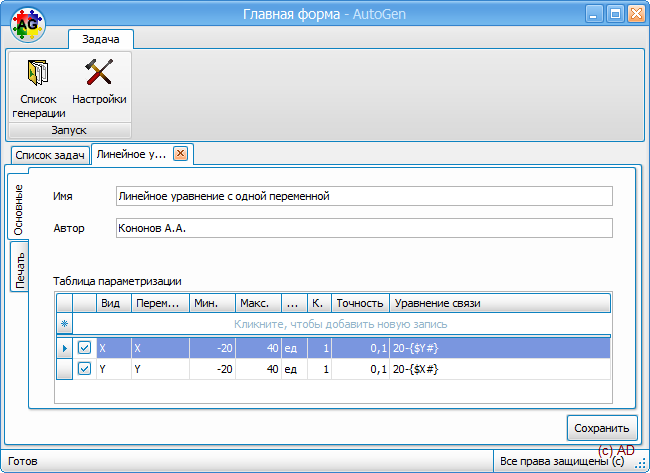


Рис 7. Редактирование задачи.

При определении таблицы параметризации важно учитывать особенности машинного счета. Так, например, при указании большого диапазона значений можно получить недостоверные результаты. Это связано с тем, что значения искомого параметра выбираются случайным образом на заданном отрезке и возможна ситуация, когда никакие значения параметров из их диапазонов не удовлетворяют условию связи. В нашем примере рассмотрено простейшее уравнение вида X + Y = 20. Результатом генерации же служит следующий текст по шаблону:

*Пусть X + Y = 20. Найти «Искомое»*

*Ответ: «Искомое» = «Результат»*

Также для перевода строки в выходном файле использовано сочетание «\\» - это стандартная команда ТеХ, осуществляющая перенос строки.

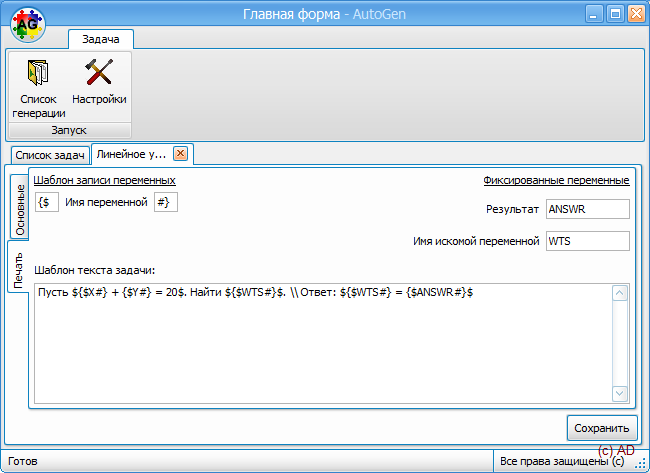


Рис . Изменение параметров печати.

В главном меню находится список доступных плагинов (Рис 9). Окно имеет вкладки для отображения плагинов – генераторов и модулей – принтеров. Здесь можно просмотреть информацию по загруженным плагинам и добавить новые.

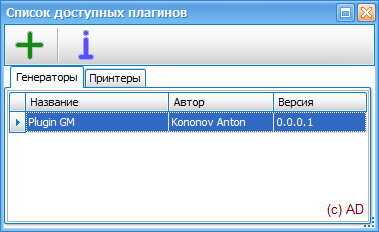


Рис . Список загруженных плагинов

Добавление плагинов производится простым копированием, можно добавлять несколько плагинов одновременно. Все изменения вступят в силу только после перезапуска программы. Это позволяет избежать возможных конфликтов в работе всей системы.

При нажатии на кнопку настройки открывается диалог настроек системы (Рис 10). Фактически здесь выбирается модуль печати по умолчанию, а также можно показать окно настроек выбранного модуля (Рис 11). В нашем модуле доступны три параметра:

* «Окно генерации» - при установке этого флажка будет отображено стандартное консольное окно с отображением процесса генерации файла PDF из данных сгенерированных задачей. Это может быть удобно во время отладки, так как не всегда текст, сформированный модулем – генератором, может быть правильно сформатирован в нужный файл. В этом случае может понадобиться вмешательство пользователя.
* «Открывать файлы» - если установить этот флажок, то после генерации каждого файла система попытается открыть его приложением по умолчанию. Обычно, для просмотра файлов PDF используют официальную бесплатную программу Adobe Reader. Эту опцию удобно применять, когда нужно сразу видеть результат.
* «Папка для сохранения» - указывает путь к каталогу, в который будут сохраняться файлы после их формирования.

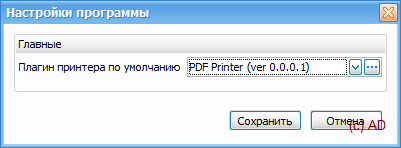


Рис . Выбор модуля печати

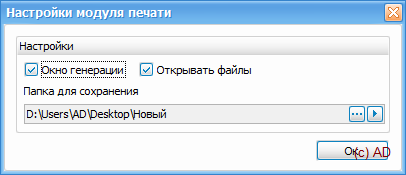


Рис 11. Настройки модуля PDF Printer

Когда все необходимые задачи сформированы и произведены нужные настройки можно переходить к генерации заданий. Этот процесс запускается из списка генерации (Рис 12), открыть его можно нажав соответствующую кнопку на панели меню. Добавлять задачи в список можно либо перетаскиванием из основного списка, либо из того же списка при помощи контекстного меню (вызывается при нажатии правой кнопки мыши).

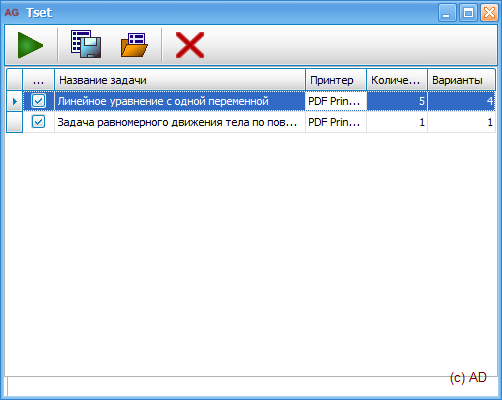


Рис 12. Список генерации задач.

На панели меню в списке расположены кнопки (слева направо):

* «Запустить» - начинает процесс генерации отмеченных Задач.
* «Сохранить список» - позволяет сохранить список в отдельный файл на диске. В дальнейшем можно просто открыть ранее собранный набор и просто заново сгенерировать задачи.
* «Открыть список» - соответственно открывает файл списка генерации.
* «Удалить запись» - удаляет выбранные записи из списка. При этом выбирать можно несколько записей одновременно.

В списке для каждой записи можно установить отдельные значения количества вариантов и количества задач в варианте. Эти числа передаются в Задачу генератор. Также в списке можно снять отметку генерации с задачи. Это позволяет, не изменяя всего списка генерировать только нужные в данный момент Задачи. При нажатии на кнопку начала генерации вид окна списка немного изменяется (Рис 13). На время генерации становятся недоступны кнопки верхнего меню и сам список. В это же время открывается панель сообщений, в которую выводится наиболее важная информация и появляются кнопки на нижней панели. При нажатии на кнопку «Скрыть» окно листа генерации скрывается, а процесс генерации отображается в главном окне. Поскольку процесс генерации может занимать довольно продолжительное время это позволит освободить пространство на рабочем столе. Рядом расположена кнопка «Прервать», она позволяет пользователю послать сигнал о необходимости прекращения процесса генерации. Как только этот сигнал будет обработан, генерация прекратится.

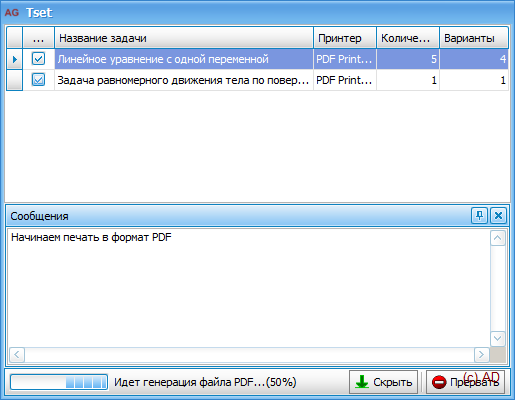


Рис 13. Окно списка в момент генерации.

Из-за того, что вся система построена на модульном принципе, после нажатия на кнопку «Прервать» и реальным прекращением процесса может пройти длительный промежуток времени. По окончании генерации список будет вновь показан, таким образом пользователь не пропустит этот момент.

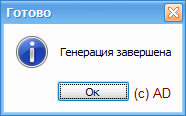


Рис 14. Сообщение об окончании генерации.

# Заключение

## Отчет по решенным и нерешенным задачам.

В результате проделанной работы мы получили работающее приложение, отвечающее современным требованиям и полностью реализующее поставленные задачи. Само приложение является оболочкой, взаимодействующей с пользователем. Его архитектура позволяет добавлять неограниченное число дополнений – генераторов. При разработке программы были учтены основные принципы модульности:

* Простота в обращении
* Универсальность
* Расширяемость

Для демонстрации механизма работы программы был разработан модуль – генератор, позволяющий генерировать простейшие примеры на основе таблиц параметризации. Особенно необходимо указать на то, что разработанный модуль не является универсальным средством генерации. Его возможности очень ограниченны. В этом модуле используются все основные возможности программы, его изучение позволит разработчикам в дальнейшем также грамотно строить функционал собственных разработок. Подробное описание некоторых возможностей также приведено внутри исходного кода программы.

Реализация форматирования информации с использованием языка разметки ТеХ позволяет разработчику создавать сколь угодно сложные оформления и форматы. Модуль печати использует всю силу компилятора ТеХ, что в совокупности с реализацией переходной спецификации TeXML дает поистине безграничные возможности. Стоит, однако, заметить, что в разработанный модуль печати со временем могут быть внесены некоторые коррективы.

## Взгляд в будущее.

Оглядываясь назад, можно увидеть огромную пропасть, разделяющую образование до и после изобретения персональных компьютеров. Технический прогресс совершил невероятный прыжок в будущее, и сейчас самое главное для остальных – это не отстать. Множество идей, двигающих процесс образования в сторону высоких технологий, так и остаются не реализованными. По большей части это происходит из-за недостатка финансирования в этой области. Эту проблему сегодня с успехом решает разработка в рамках лицензии бесплатного программного обеспечения.

Имея основательную базу и поддержку научного сообщества можно очень быстро добиться невероятных результатов. Автор надеется, что эта работа послужит той самой базой, толчком к новому витку развития процесса образования в современном информационном мире.

# Список литературы

1. **Free Software Foundation, Inc** GNU GENERAL PUBLIC LICENSE [В Интернете]. - 2007 r.. - http://www.gnu.org/licenses/gpl-3.0-standalone.html.
2. **Fullerton** Steepest Descent or Gradient Method [В Интернете]. - 2008 r.. - http://math.fullerton.edu/mathews/n2003/GradientSearchMod.html.
3. **Ашкинази Леонид Александрович** Генератор задач по физике [В Интернете] // Информационные технологии в образовании. - 2007 r.. - http://ito.edu.ru/2007/Moscow/II/1/II-1-6899.html.
4. **Бочканов Сергей** Библиотека алгоритмов [В Интернете]. - 2008 r.. - http://alglib.sources.ru.
5. **Википедия** Microsoft.NET [В Интернете] // Википедия. - 28 Май 2008 r.. - http://ru.wikipedia.org/wiki/Microsoft.NET.
6. **Гайдышев И** Анализ и обработка данных: специальный справочник [Книга]. - Санкт Петербург : Издательский дом "Питер", 2001.
7. **Дубовцев А В** Microsoft.NET в подлиннике [Книга]. - Санкт Петербург : БХВ-Петербург, 2004.
8. **Ионов А Ю, Копылов И А и Жуков Д О** Генератор задач по физике [В Интернете] // Московская государственная академия приборостроения и информатики, Россия. - Ноябрь 2000 r.. - http://nit.miem.edu.ru/2003/tezisy/articles/50.htm.
9. **Хейфец Илья** Архитектура .NET [В Интернете] // Interface.Ru. - 11 Ноябрь 2002 r.. - http://www.interface.ru/fset.asp?Url=/microsoft/arx.htm.

# Приложение 1. Описание интерфейсов программы

namespace AutoGen.I

{

/// <summary>

/// Интерфейс параметров генерации

/// </summary>

public interface IAutoGenParameters

{

/// <summary>

/// Количество заданий в варианте

/// </summary>

int CountInVariant { get; set; }

/// <summary>

/// Количество вариантов

/// </summary>

int Variants { get; set; }

/// <summary>

/// Нужен ли ответ на задачу

/// </summary>

bool NeedAnswer { get; set; }

/// <summary>

/// Название задачи

/// </summary>

string TaskName { get; set; }

}

/// <summary>

/// Интерфейс обеспечивающий чтение и запись

/// текстовых данных в конфигурационный файл

/// </summary>

public interface IAutoGenDBSettings

{

/// <summary>

/// Сохранить данные

/// </summary>

/// <param name="set">Имя набора данных</param>

/// <param name="name">Название</param>

/// <param name="value">Данные</param>

void SaveValue(string set, string name, string value);

/// <summary>

/// Получить данные

/// </summary>

/// <param name="set">Имя набора данных</param>

/// <param name="name">Название</param>

/// <returns>Данные</returns>

string GetValue(string set, string name);

}

/// <summary>

/// Интерфейс главного приложения

/// </summary>

public interface IAutoGenApplication

{

/// <summary>

/// Предоставляет объект доступа к настройкам

/// </summary>

IAutoGenDBSettings MainDBSettings { get; }

/// <summary>

/// Главная форма программы

/// </summary>

object MainForm { get; }

/// <summary>

/// Директория приложения

/// </summary>

string MainApplicationDir { get; }

/// <summary>

/// Поддиректория \ТЕХ

/// </summary>

string MainTexDir { get; }

/// <summary>

/// Расположение портативной версии ТеХ

/// </summary>

string MainTexPortDir { get; }

/// <summary>

/// Расположение файлов данных

/// </summary>

string MainDataDir { get; }

}

/// <summary>

/// Интерфейс плагина - дополнения

/// </summary>

public interface IAutoGenPlugin

{

/// <summary>

/// Версия

/// </summary>

Version PluginVersion { get; }

/// <summary>

/// Автор

/// </summary>

string Autor { get; }

/// <summary>

/// Короткое название

/// </summary>

string PluginName { get; }

/// <summary>

/// Произвести инициализацию объекта плагина

/// </summary>

/// <param name="autoGenApp">Главное приложение</param>

void InitPlugin(IAutoGenApplication autoGenApp);

/// <summary>

/// Отобразить информацию о плагине

/// </summary>

void ShowAbout();

/// <summary>

/// Получить уникальный идентификатор плагина

/// </summary>

Guid GUID { get; }

}

/// <summary>

/// Интерфейс дополнения - генератора

/// </summary>

public interface IAutoGenGenerator : IAutoGenPlugin

{

/// <summary>

/// Получить новый объект Задачи

/// </summary>

/// <param name="taskName">Имя задачи</param>

/// <returns>Объект Задачи для редактирования</returns>

IAutoGenTask CreateTaskInstance(string taskName);

/// <summary>

/// Дополнительные данные плагина

/// </summary>

Object GeneratorData { get; set; }

}

/// <summary>

/// Интерфейс экземпляра Задачи для генерации

/// </summary>

public interface IAutoGenTask : ISerializable

{

/// <summary>

/// Название

/// </summary>

string TaskName { get; set;}

/// <summary>

/// Описание

/// </summary>

string TaskDescription { get; set; }

/// <summary>

/// Объект элемента управления

/// </summary>

ITaskControl TaskPropertiesControl { get;}

/// <summary>

/// Произвести генерацию задачи

/// </summary>

/// <param name="Parameters">Параметры</param>

/// <param name="Worker">Обработчик</param>

/// <returns>Объект формата TeXML</returns>

TeXMLDoc GenerateTask(IAutoGenParameters Parameters, IAutoGenWorker Worker);

}

/// <summary>

/// Интерфейс элемента управления задачей

/// </summary>

public interface ITaskControl

{

/// <summary>

/// Основной элемент, управляющий данными задачи

/// </summary>

Control InnerControl { get; }

/// <summary>

/// Функция вызывается перед закрытием вкладки

/// </summary>

/// <param name="sender">Главная форма</param>

/// <param name="e">Аргументы</param>

void ParentTabClosing(object sender, CancelEventArgs e);

/// <summary>

/// Событие происходяшие в момент сохранения задачи

/// </summary>

event TaskChangeEventHandler TaskSaved;

/// <summary>

/// Событие происходящие в момент изменения задачи

/// </summary>

event TaskChangeEventHandler TaskChanged;

}

/// <summary>

/// Интерфейс модуля принтера

/// </summary>

public interface IAutoGenPrinter : IAutoGenPlugin

{

/// <summary>

/// Запустить печать документа

/// </summary>

/// <param name="TeXDocument">Документ в формате TeXML</param>

/// <param name="Worker">Обработчик</param>

/// <param name="Parameters">Параметры</param>

void Print(TeXMLDoc TeXDocument, IAutoGenWorker Worker,

IAutoGenParameters Parameters);

/// <summary>

/// Отобразить настройки модуля

/// </summary>

void ShowProperties();

}

/// <summary>

/// Интерфейс потокового обработчика заданий

/// </summary>

public interface IAutoGenWorker

{

/// <summary>

/// Сообщить о выполнении

/// </summary>

/// <param name="ProgressPercent">Процент выполнения</param>

/// <param name="ProgressCaption">Стадия выполнения</param>

void ReportProgress(int ProgressPercent, string ProgressCaption);

/// <summary>

/// Записать сообщение в лог

/// </summary>

/// <param name="line">Строка сообщения</param>

void WriteOutputLine(string line);

/// <summary>

/// Послать сигнал отмены выполнения

/// </summary>

void CancelProgress();

/// <summary>

/// Происходит, когда кто-либо из участников

/// посылает сигнал отмены

/// </summary>

event EventHandler CancelSend;

/// <summary>

/// Признак того, что запущена отмена

/// </summary>

bool IsCanceled { get; }

}

}

# Приложение 2. Лицензия GNU GPL

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
* b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
* c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
* d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
* b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
* c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
* d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
* e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
* b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
* c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
* d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
* e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
* f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License

along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

This is free software, and you are welcome to redistribute it

under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

1. http://www.gnu.org/licenses/gpl.html [↑](#footnote-ref-2)
2. http://ru.wikipedia.org/wiki/XML [↑](#footnote-ref-3)
3. http://getfo.org/texml/spec.html [↑](#footnote-ref-4)
4. http://sourceforge.net и http://code.google.com [↑](#footnote-ref-5)